Triangle count estimation and label prediction over uncertain streaming graphs

by

Ipsita Mohanty

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis aims to integrate the notions of uncertainty with graph stream processing, presenting probabilistic models to enhance real-time analytical capabilities in graph database systems. These systems are crucial for managing interconnected data in various domains, such as social networks, traffic networks, and genomic databases, where data often contains incomplete or probabilistic connections that complicate processing and analysis.

We develop and validate two main methodologies: a martingale-based approach for approximating triangle counts in edge uncertain streaming graphs and a Graph Neural Network (GNN)-based method for dynamic label prediction in attribute uncertain streaming graphs. Both methods demonstrate robust performance in handling dynamic and uncertain data, thus opening new avenues for future research in expanding the scope of graph-based analytics. This work lays the groundwork for future developments in uncertain graph processing, suggesting pathways to refine these approaches and explore new applications in dynamic environments.

Acknowledgements

I extend my deepest gratitude to Prof. Tamer Özsu, whose unwavering support and mentorship have been pivotal throughout my academic journey. Prof. Özsu not only taught me invaluable skills in time management and research methodology but also significantly boosted my confidence and motivation. His vast knowledge served as a constant resource, enriching my understanding of big data and its possibilities. His open-mindedness, kindness, wisdom, intelligence, and thoughtfulness have profoundly shaped my professional and personal growth. I also extend my thanks to Prof. Lukasz Golab and Prof. Grant Weddell for their willingness to read and review my thesis.

Being a member of the Data Systems Group has been a privilege and a rich source of learning. Every moment has been a learning opportunity, from seminars and weekly lunch talks to engaging with researchers and contributing to the lab committee. A heartfelt thank you goes to my friends and colleagues, who have been my support system throughout this journey - Kriti, Aida, Sairaj, Shubhankar, Xiangru, Pulkit, Lian, Max, Chao, Zeynep, Shirley, and countless others whose names I regrettably cannot mention individually.

I also owe a profound debt of gratitude to my family, whose unwavering support has been my foundation throughout this journey. To my parents, whose love and guidance have been my constant compass, thank you for instilling in me the values of hard work and perseverance. To my grandparents, whose stories and wisdom have enriched my life and inspired me to aim high, your experiences have taught me more than any book could. And to my brother, Shreyarth, thank you for the endless encouragement and for always believing in me even when doubts clouded my path. Your collective support and sacrifices have not only shaped who I am but have also made this academic achievement possible. I am eternally grateful and proud to share this success with you all.

Dedication

This thesis is dedicated to my parents, Kuntala and Chittaranjan, and my brother, Shreyarth.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures ix

List of Tables xi

1 Introduction 1

1.1 Types of Uncertainty in Graphs 4

1.2 Research Challenges 5

1.3 Contributions and Organization 6

2 Background & Related Works 9

2.1 Uncertain Graphs 9

2.1.1 Possible World Semantics 9

2.1.2 Uncertain Graph Management and Mining 12

2.2 Overview of Research on Uncertain Graphs 13

2.2.1 Reliability Queries . . . 13
2.2.2 Graph Pattern Matching . . . 14
2.2.3 Influence Maximization . . . 15
2.3 Streaming Graphs . . . 16
2.3.1 Unboundedness and Real-Time Processing . . . 17
2.3.2 Graph Mining Problems . . . 18

3 Edge Uncertain Streaming Graphs **21**
3.1 Edge Uncertain Streaming Graph Model . . . 22
3.2 Estimating the Number of Active Vertices . . . 22
3.3 Active vertex estimation example . . . 28
3.4 Estimating the Number of Triangles . . . 29
3.4.1 Doob or Exposure Martingales [4] . . . 31
3.4.2 Edge Exposure Martingale . . . 34
3.5 Triangle count estimation example . . . 39
3.6 Experimental Evaluation . . . 39
3.6.1 Data Generation . . . 40
3.6.2 Experimental Platform . . . 41
3.6.3 Accuracy Experiments . . . 42
3.6.4 Latency Experiments . . . 45

4 Label Uncertain Streaming Graphs **47**
4.1 Definitions and Target Measures . . . 48
4.2 Label Prediction . . . 50
4.2.1 Embedding generation . . . 51
4.2.2 Label Classification . . . 51
4.3 Experimental evaluation . . . 53
4.3.1 Experimental Platform . . . 54
4.3.2 Accuracy Experiments . . . 54
4.3.3 Latency Experiments . . . 55

5 Conclusion and Future Work **61**

5.1 Future Work 62

References **63**

List of Figures

1.1 Interaction network of Mic17 obtained from the STRING database. All interactions (i.e., uncertain edges) are derived from experimental evidence . 2

2.1 Set of possible worlds of the Uncertain Graph with its probability of occurrence 11

2.2 Graph Snapshot $\mathcal{G}_1$ corresponding to $[t_0, t_4]$ 14

3.1 Streaming graph 25

3.2 Graph Snapshot $\mathcal{G}_1$ corresponding to $[t_0, t_4]$ 25

3.3 Graph Snapshot $\mathcal{G}_2$ corresponding to $[t_2, t_6]$ 25

3.4 Graph Snapshot $\mathcal{G}_3$ corresponding to $[t_4, t_8]$. Notice that the node B has expired. 26

3.5 Graph Snapshot $\mathcal{G}_4$ corresponding to $[t_6, t_{10}]$ 26

3.6 Example Graph 29

3.7 The edge exposure martingale with $n = m = 3$, representing the number of distinct colours required for acceptable vertex colouring. The edges are exposed in the order "bottom, left, right". The value of X_i are given by tracing from the central node to the leaf. 37

3.8 Degree distribution in Wikipedia-election dataset 41

3.9 Degree distribution in Facebook dataset 42

3.10 log-log graph of degree distribution in Wikipedia-election dataset 43

3.11 log-log graph of degree distribution in Facebook dataset 44

3.12 Degree distribution in our synthetic dataset 45

3.13 log-log graph of degree distribution in our synthetic dataset 46

4.1 The GraphSAGE aggregate approach [23] 48

4.2 The message passing mechanism. Image source: CS224 slides Stanford University 53

4.3 log-log graph of degree distribution in the synthetic dataset with 5 labels . 54

4.4 Edge label distribution in the synthetic dataset with 5 labels 55

4.5 Accuracy of prediction algorithm with 5 labels 56

4.6 log-log graph of degree distribution in the synthetic dataset with 10 labels 57

4.7 Edge label distribution in the synthetic dataset with 10 labels 57

4.8 log-log graph of degree distribution in the synthetic dataset with 10 labels 58

4.9 Edge label distribution in the synthetic dataset with 10 labels 58

4.10 Accuracy of prediction algorithm with 10 labels 59

4.11 Accuracy of prediction algorithm with 15 labels 60

List of Tables

3.1 Accuracy results 46

Chapter 1

Introduction

Graph database systems (GDBMS) have been gaining immense popularity over the last few years. These systems make it easy to model complex interconnected data like social networks, information networks, traffic networks, genome databases, and medical and government records. GDBMSs can also efficiently organize image [45] and video [26] data. Many graphs, such as web and social networks, can have trillions of edges and are often dynamic – i.e., their structure changes over time. Graph database systems thus become essential for the practical storage, processing, and analysis of these large, evolving, and complex databases.

In many real-life scenarios, researchers and practitioners often encounter uncertain data. This uncertainty often stems from the inherent limitations in our observations, which may often only capture the partial aspects of complex realities we seek to understand. Moreover, the presence of noise in the observations further increases the complexity. These errors and inaccuracies can often blur the actual state of affairs.

Consider the example of medical data. While some diseases may exhibit precise and deterministic relationships between symptoms and underlying causes, many others present a more ambiguous landscape. In such cases, medical practitioners must navigate through a maze of potential diagnoses, considering various factors and probabilities before arriving at a conclusion.

This type of uncertainty pervades various fields, including but not limited to cell biology, finance, climate science, and social networks. In Protein-Protein Interaction (PPI) networks (see Figure 1.1), connections between two protein molecules denote possible interactions, typically identified through often uncertain and noisy experiments. This uncer-

tainty can be effectively captured by assigning an uncertainty or, in more concrete terms, a probability to the existence of each connection. Likewise, in social networks, the probability of having a connection or "edge" between two users can serve as a proxy for the likelihood of interaction or influence exerted by one user on the other. Thus, uncertainty is intrinsic to graph data, which can arise due to a multitude of factors like inaccuracies in measurements [3], unreliability and ambiguity of data sources [10], imprecise prediction models [54, 2], or deliberate alterations made for privacy purposes [8]. In all these cases, data can be more effectively represented as an uncertain graph, that is a graph whose vertices, edges, or attributes are accompanied by a probability of existence or validity. This probabilistic framework not only helps acknowledge the inherent uncertainties within data but also facilitates a more nuanced understanding of the underlying uncertainties inherent in various real-world phenomena.

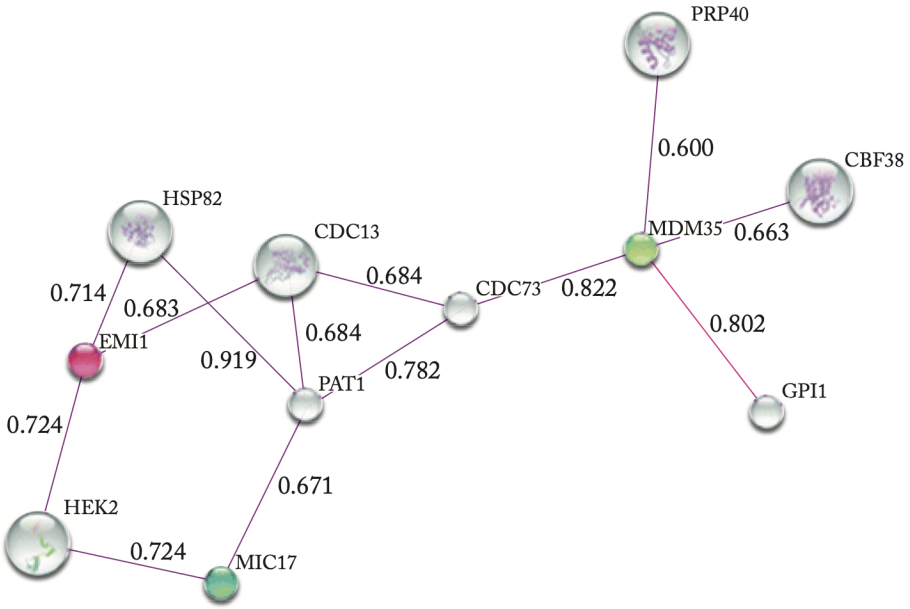


Figure 1.1: Interaction network of Mic17 obtained from the STRING database. All interactions (i.e., uncertain edges) are derived from experimental evidence

Further, in various modern contexts like product transactions, online feeds, and social networking, graphs continuously "emerge," i.e., entities and connections arrive over time. These are called *streaming graphs*, and their notable feature is that they are unbounded in nature, and it is impractical for processing systems to have a "holistic" view of the entire

graph at any particular time. The ongoing evolution of these graphs leads to unpredictable stream durations, varied data distributions, and irregular arrival rates. Update events are focused on the most recent graph topology, resulting in non-uniform arrival patterns and the possibility of out-of-sequence data arrival from multiple sources and transmission delays beyond the processing unit's control.

Thus, merging the notions of uncertainty and streaming is the natural next step. Combining streaming and uncertain graph concepts offers significant advantages, particularly in domains where data evolves continuously and carries inherent uncertainties. Some of the reasons why combining these concepts might be beneficial are the following:

- **Real-time decision making:** Streaming graphs depict evolving data, while uncertain graphs factor in uncertainties like missing information or probabilistic relationships. This merger enables real-time decision-making with a nuanced understanding of uncertainties, thus enhancing outcome reliability.
- **Improved analytics and insights:** Integrating uncertainty into streaming graphs allows analytics to consider the probability of events or relationships, yielding richer insights. For instance, in fraud detection, understanding uncertainty in transaction patterns aids in nuanced risk assessments.
- **Enhanced predictive modelling :** By incorporating the dynamic nature of data and its uncertainties, merging these concepts improves predictive modelling. This is particularly useful in scenarios like predictive maintenance, where models must adapt to new data and uncertainties in sensor readings or environmental conditions.
- **Robustness in dynamic environments:** Combining streaming and uncertain graph concepts helps build robust models that handle rapid environmental changes and noisy or incomplete data. This ensures reliable insights despite the dynamic nature of the data.
- **Optimized resource allocation:** Understanding real-time network states and uncertainties in demand or supply improves resource allocation in applications like network traffic management or supply chain optimization, leading to more efficient operations and anticipation of potential issues.
- **Facilitating complex event processing:** Integration of these concepts enhances complex event processing systems, enabling them to react to evolving patterns while considering uncertainties, thereby offering a comprehensive event processing solution.

Inspired by the mentioned applications, this thesis centres on integrating streaming graphs and uncertain graphs. It examines classical issues in streaming data mining and explores how embracing uncertainties enables advanced analysis. The outcome encapsulates the nature of dynamic data and inherent uncertainties, facilitating informed decisions and resilient applications across diverse domains.

1.1 Types of Uncertainty in Graphs

In an *uncertain* or *probabilistic* graph, uncertainty can be associated with any one or multiple of the following components:

- **Edge uncertainty:** Uncertainty in edges is a well-studied concept in uncertain graph research, focusing on the probability of a connection existing between two vertices. This uncertainty arises from various sources, including noise, measurement errors, inference processes, and predictive modelling. Additionally, the probability associated with an edge can encapsulate a range of different edge characteristics [6]. As an example, in a social media context, a popular musician may have a stronger influence on their followers regarding music and style updates, while a well-known political figure might have a more significant impact with tweets focused on political issues among their audience.

- **Vertex uncertainty:** The concept of vertex uncertainty has been examined in relation to device networks and the execution of graph pattern-matching queries. The most common way to understand vertex uncertainty is by associating it with the likelihood that the vertex actually exists. This interpretation helps assess the reliability of vertices within networks, especially when analyzing complex network structures or addressing pattern-matching queries. For example, [53] introduced the notion of identity uncertainty, that is, uncertainty about whether each real-world entity is represented by one or multiple vertices in the graph.

- **Attribute/Label uncertainty:** Uncertainty regarding an attribute or label pertains to the ambiguity in the attribute values linked to the vertices or edges. This concept finds application in various graph-related contexts, including graph pattern matching, the creation and handling of uncertain query graphs, and the execution of queries on RDF (Resource Description Framework) data. Such uncertainty plays a critical role in interpreting and processing data accurately within these applications.

The existing work on these types of uncertainties focuses on static graphs, while the focus in this thesis is on the emergence of these concepts in streaming graphs. In particular, we investigate edge uncertainty and label uncertainty. We delve into one extensively studied problem within each realm, specifically **Approximate Triangle Counting** for edge uncertainty and **Label Prediction** for label uncertainty. Further details on these topics will be explored in subsequent chapters.

1.2 Research Challenges

The main focus of this thesis is *uncertain streaming graphs*: a dynamic and probabilistic framework where edges and/or attributes are imbued with probabilities that reflect the inherent uncertainty in their existence and characteristics. In this context, the system receives timestamped payloads (defined precisely in Chapter 2), capturing the temporal evolution of the graph structure and properties. The uncertainty information is assumed to be associated with each payload, and we examine and analyze the structural patterns of these payloads aggregated over a desired timeframe. By delving into uncertain streaming graphs, this thesis aims to inspire novel methodologies and algorithms capable of effectively analyzing and extracting meaningful insights from streaming data in real-time scenarios.

Streaming graph processing is an essential area of research that addresses the processing and analysis of graph data in real-time or near-real-time. As the data continuously evolves, streaming graph processing poses unique challenges compared to static graph processing. Adding the uncertainty to the streaming graph problem adds another layer of complexity to the model. Here are some of the key research challenges in this field:

- **Unboundedness:** Streaming data presents a unique challenge due to its potentially infinite nature, constantly arriving without a predetermined endpoint. This continuous influx thus requires innovative approaches to managing, storing, and processing the data, as traditional batch processing methods become impractical in this context. One effective strategy in response to this challenge is to employ a time-based window analysis technique on the graph stream. By implementing such an approach, we can effectively segment the data into finite chunks based on time intervals, allowing for more efficient processing and analysis of the streaming data. This method enables real-time insights and decision-making, which is crucial in dynamic environments where getting real-time results is paramount.

- **Modelling uncertainties:** A key challenge lies in formulating effective methodologies for modelling uncertainty within graph representations. Firstly, for effective encapsulation of inherent uncertainties and interdependencies in graph streams, a robust probabilistic framework for the graph stream must be established. Secondly, within the context of a streaming system, it is challenging to continually update the model to incorporate the latest information as new data arrive, as this may result in a significant computational overhead, particularly in scenarios requiring extensive model retraining or Bayesian updating procedures. Further, guaranteeing the consistency and validity of the model in response to new incoming data becomes increasingly complex. This thesis shows that a simple Bayesian update analogous to the **PageRank** algorithm works well for simple cases.
- **Benchmarking and evaluation:** Since analyzing uncertain streaming graphs presents a fresh challenge, it is crucial to establish standardized benchmarks and evaluation criteria to gauge the effectiveness and precision of algorithms designed for uncertain stream processing. While conducting tests on static uncertain graph frameworks offers insights, devising evaluation strategies and metrics tailored to this specific issue is imperative. This ensures a more comprehensive understanding of algorithm performance in dynamic, real-time environments.

Addressing these challenges requires interdisciplinary approaches that combine graph theory, probability and statistics, algorithm design, data management strategies, and efficient computation models.

1.3 Contributions and Organization

As previously mentioned, there are numerous methods to integrate uncertainties into streaming data. Specifically, within this thesis, our attention is centred on edge and label/attribute uncertainties. To elaborate further, we concentrate on addressing a single problem within each domain and present a thorough framework for tackling these challenges. Brief details of the problems are outlined below:

- **Approximate triangle counting in edge uncertain streaming graphs**
  Counting triangles plays a crucial role in graph processing. Various graph models have extensively employed it as a network analysis metric (e.g., calculating transitivity ratios). Further, it has various practical applications such as spam detection, finding hidden thematic structures within the Web, and link recommendations.

Given the probabilistic nature of our model, we leverage graph theoretic principles and probabilistic methodologies to establish an asymptotically tight estimate of the triangle count within a defined temporal window. This approach allows for a rigorous characterization of the triangular connectivity patterns within evolving graph structures, thus facilitating deeper insights into their dynamics and functionalities. Further, as there is no actual "counting" involved, we get faster results than deterministic models. This thesis allows for trading off running time against exactness. Furthermore, it enables approximating the number of triangles in a graph that does not completely fit in the main memory.

- **Label prediction in attribute uncertain streaming graphs**
  Label prediction in graph streams can be highly complex due to the dynamic nature of graphs and the need to capture interdependencies between labels. In general, for static graphs, machine learning frameworks in a semi-supervised context can provide significant flexibility. Unlike strictly supervised or unsupervised learning, graph-based machine learning can utilize the structure and connectivity of the graph to infer labels, exploiting both labelled and unlabeled data within a connected framework. However, as streaming systems require real-time/near-real-time computations, using these computation-intensive ML models for link prediction becomes impractical.

  This thesis enhances the prediction process by leveraging the uncertainty model to achieve accelerated outcomes. Given the inherently probabilistic nature of predictions, prioritizing speed does not significantly compromise accuracy, enabling tear real-time results while conserving computational resources.

The remainder of the thesis is organized as follows -

- Chapter 2 presents background information and summarizes the related existing work on uncertain graph analysis and stream management and processing systems.
- Chapter 3 establishes formal foundations for representing the uncertain graph model targeted in this thesis. It tackles the problem of approximate triangle counting, as described above, and gives rigorous proofs ascertaining the tightness and correctness of the calculated bound.
- Chapter 4 focuses on the design and implementation of a label prediction framework that extends the probabilistic model and uses it to predict new edges and their attributes. It employs a simple GNN model to analyze subgraph patterns and predicts edge attributes accordingly.

- Finally, Chapter 5 summarizes the experiments that validate the correctness of the models discussed in Chapter 3 and Chapter 4 and presents their results. We also discuss some limitations of our model and directions for future research.

Chapter 2

Background & Related Works

This chapter is structured around three principal sections. Section 2.1 and Section 2.2 provide an overview of the core concepts and the necessary background on uncertain graphs and discuss existing relevant literature in this field. Then Section 2.3 introduces the preliminary definitions related to streaming graphs, examining the existing work in this domain that is relevant to this thesis.

2.1 Uncertain Graphs

Before delving into the core concepts, we first define the uncertain graph model used in the thesis. These definitions form the basis of the discussions and methodologies that follow.

2.1.1 Possible World Semantics

In recent years, there has been a notable increase in research dedicated to exploring various aspects of uncertain graph data. This includes its representation, storage, query processing, analysis, and mining. A widely adopted method for expressing the semantics of *edge uncertain graphs* (EUG) is through the *possible world model* [42], [27]. This model suggests that an EUG represents a probability distribution across all potential scenarios or possible worlds. Each possible world represents a deterministic graph where any particular edge could potentially exist. For example, an EUG with m edges yields 2^m possible deterministic graphs, which are derived by sampling independently each edge with its corresponding probability.

Definition 1 (Graph). *An undirected graph $G = (V, E)$ is a collection of $n = |V|$ vertices, and $E \subseteq V \times V$ is a set of $m = |E|$ edges.*

Definition 2 (Edge Uncertain Graph). *A EUG $\mathcal{G}$, is a tuple (V, E, p) where V represents the set of n vertices in the graph, $E \subseteq V \times V$ is the set of m edges, and the function $p : E \to [0, 1]$ assigns to each edge in E its corresponding probability or likelihood of existence.*

Definition 3 (Possible World Graph). *A deterministic graph, $G \subseteq \mathcal{G}$, can be described by the pair (V, E_G), where E_G is a subset of E. This is one of the 2^m possible worlds that can be generated by $\mathcal{G}$. The probability of sampling this specific graph G from $\mathcal{G}$ is given by-*

$$Pr(G) = \prod_{e \in E_G} p(e) \prod_{e' \in E \setminus E_G} (1 - p(e'))$$

Figure 2.1 illustrates an EUG with five vertices and seven edges and three of its $2^7 = 128$ possible worlds with their sampling probability.

One may note that the probabilities of the existence of all possible worlds derived from an EUG are independent of each other, and they add up to 1.

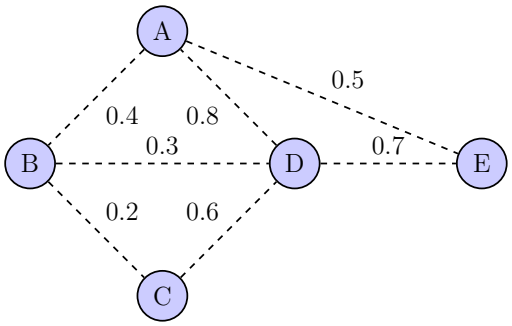
The semantic framework outlined for EUG can also be applied to *label uncertain graphs* (LUG).

Definition 4 (Labelled Graph). *A labelled graph G can be represented as a four-element tuple (V, E, Σ, L) where V is the set of vertices, $E \subseteq V \times V$ is the set of edges, Σ is a set of labels, $L : E \to \Sigma$ is a function that assigns labels to edges.*

Definition 5 (Label Uncertain Graph). *A label-uncertain graph, denoted as $\mathcal{G}$, can be defined by a quintuple (V, E, Σ, p_e, p_l). In this tuple, V is the collection of n vertices within the graph, $E \subseteq V \times V$ constitutes the set of m edges, and Σ denotes the collection of possible labels. The function $p_e : E \to [0, 1]$ assigns to each edge in E its corresponding probability or likelihood of existence and the function $p_l : E \times \Sigma \to [0, 1]$ maps each edge and its associated label to a probability value, indicating the likelihood of each edge-label combination.*

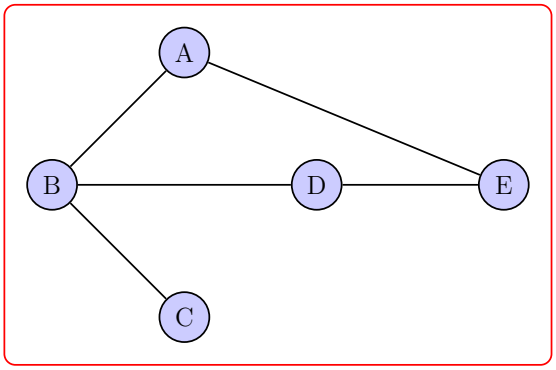
Definition 6 (Possible World Graph). *A possible world graph $G = (V_G, E_G, \Sigma_G, L)$ is an instantiation of an uncertain graph $\mathcal{G} = (V, E, \Sigma, p)$ where $V_G \subseteq V$, $E_G \subseteq E$, $\Sigma' \subseteq \Sigma$ and $L : E_G \to \Sigma_G$ is a function that assigns labels to edges. The probability of sampling this specific graph G from $\mathcal{G}$ is given by -*

$$Pr(G) = \prod_{e \in E_G} p_e(e) p_l(e, L(e)) \prod_{e' \in E_G \setminus E} (1 - p_e(e'))$$

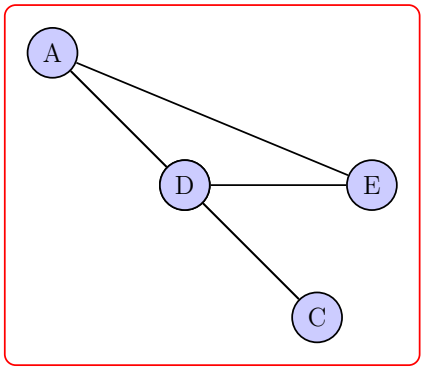


Uncertain Graph $\mathcal{G}$

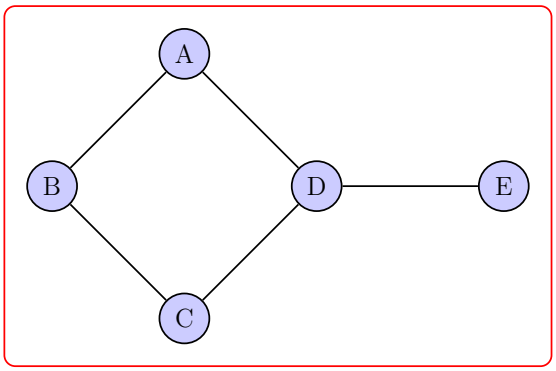
Possible Worlds of $\mathcal{G}$



$Pr(G) = 0.0084$



$Pr(G) = 0.168$



$Pr(G) = 0.0269$

Figure 2.1: Set of possible worlds of the Uncertain Graph with its probability of occurrence

Definition 7. *$PWG(\mathcal{G})$ is used to denote the set of all possible world graphs derived from an EUG or LUG $\mathcal{G}$.*

2.1.2 Uncertain Graph Management and Mining

In the domain of uncertain graphs, a considerable amount of research and development activity has focused on addressing the challenges associated with its mining and management. Various tasks, such as subgraph similarity search [52], the evaluation of graph reliability to ascertain reachable nodes from given query points [30], frequent subgraph mining [57], reliable clustering [36] have been of particular interest. The possible-world model has been widely used to represent the indeterminate nature of these graphs. Nonetheless, this approach is not without its drawbacks, chiefly the considerable computational load it imposes. The root of the challenge lies in the rapid escalation of potential deterministic graphs that emerge as the uncertain graph expands - in essence, an uncertain graph with m edges could present 2^m possible worlds, and adding even one extra edge means this number doubles. Such vast possibilities render the tasks of query processing and data mining on uncertain graphs notably infeasible due to their extensive computational demands.

To manage this computational challenge and streamline the analysis of uncertain graphs, sampling methods [33] have emerged as a powerful solution. Sampling approaches involve selecting a representative subset of possible worlds to approximate the overall structure and behaviour of the uncertain graph. This approximation allows for a more practical approach to processing and mining while reducing the need for exhaustive computation across every potential graph configuration. Through these techniques, researchers and analysts can achieve a balance between computational feasibility and the precision of results derived from uncertain graph data.

Among the various sampling techniques available, the Monte Carlo method is notably the most widely utilized. It involves selecting a potential instance G from the set of all possible worlds $PWG(\mathcal{G})$, based on its probability $Pr(G)$. The variance of the sampled distribution is then reduced by leveraging the particular constraints of the given problem [52]. Nonetheless, the Monte Carlo method is not without its limitations. For instance, the sampling time for a possible world in a graph containing m edges is $O(m)$. Furthermore, to maintain the approximation error ϵ below a certain threshold δ with a high confidence level, it necessitates sampling of $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$ possible worlds, where $0 < \epsilon, \delta < 1$.

2.2 Overview of Research on Uncertain Graphs

Uncertain graphs have become a significant focus for researchers due to their wide-ranging applications in representing ambiguous interconnected data. Khan et al. [31] have published an extensive text on this subject. Uncertain graphs are increasingly prevalent in new applications, capturing the interest of both the database and data mining communities. Simple tasks, like reachability and shortest path queries, are $\#P$ complete in the uncertain graph domain. Additionally, uncertain networks see the development of intricate queries and analytics, including pattern recognition, information spread, and influence maximization. This section will explore various problem types associated with uncertain graphs and review existing research in the area.

2.2.1 Reliability Queries

A fundamental problem that has been extensively studied in the context of uncertain graphs is the problem of $s-t$ reliability, which determines the likelihood $R(s,t)$ that a target node t can be reached from a source node s. This task has been proven to be $\#P-$hard, which is why various efficient algorithms based on sampling and indexing have been developed. This discussion includes some of these methods from other studies.

- **Monte Carlo Sampling**
  This basic sampling method, as outlined in [20], involves sampling K out of the $2^{|E|}$ possible world graphs from the uncertain graph $\mathcal{G}$ based on the probability associated with the existence of independent edges. Here, the authors use a straightforward hit-and-miss approach to calculate the estimator $\hat{R}(s-t)$ for $R(s-t)$ by taking the mean of the reachability score on each of the sampled graphs. Further, it has been proven that this calculated estimator is indeed an unbiased estimator, i.e., $E(\hat{R}(s,t)) = R(s,t)$, and its variance follows a Binomial distribution [27, 20]. In Chapter 3, we more or less adopt a similar approach with streaming graphs, applying martingale models to show that our estimator is unbiased and adheres to the law of large numbers.

- **Indexing via BFS Sharing**
  Building on Monte Carlo sampling, Zhu et al. [55] developed an offline method to generate K possible worlds from the uncertain graph $\mathcal{G}$. This method involves using a compact bit-vector structure, as shown in Figure 2.2, which minimizes storage

overhead. This structure maintains a single graph, assigning a bit vector of size K to each edge. Each bit in this vector indicates the presence of the edge in a corresponding sampled graph. This approach of sharing BFS has the same variance as basic Monte Carlo sampling but significantly reduces the time required for online $s-t$ reliability estimation by preparing the possible worlds offline.

Expanding on these concepts, additional sampling techniques have been introduced to reduce both storage demands and the variance of the reliability score estimator. Jin et al. [27] introduced a recursive sampling method that uses a divide-and-conquer strategy to create the possible worlds. This approach was further refined by Kempe et al. [29], who developed a dynamic programming method that uses a sampling table at the bottom to facilitate memoization, making the algorithm iterative and thus reducing the size of the recursive stack.

Numerous other strategies for computing the reliability score are well-documented in [28], which provides a comprehensive comparison of various algorithms and their performances. According to this paper, no single approach is superior; each has its trade-offs, including factors such as Relative Error, Running Time, and Memory Usage, with different algorithms exhibiting varying efficiencies in these areas.

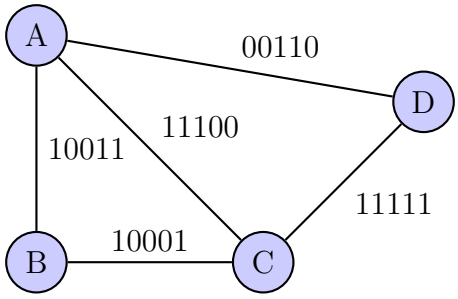


Figure 2.2: Graph Snapshot $\mathcal{G}_1$ corresponding to $[t_0, t_4]$

2.2.2 Graph Pattern Matching

A considerable amount of research has been devoted to finding efficient solutions for pattern matching over uncertain graphs. Unlike reachability or shortest-path queries, which focus on reachability and path between two specific vertices, pattern-matching queries consider the connections among groups of vertices.

More formally, the graph pattern matching problem is defined as follows. Given a graph pattern query q with n vertices $\{v_1, \ldots, v_n\}$ and an uncertain graph $\mathcal{G}$, and a pre-defined threshold ϵ, the objective is to retrieve all sets of vertices $S = \{u_1, \ldots, u_n\}$ in $\mathcal{G}$ such that the pattern matching probability of existence of S in $\mathcal{G}$ is at least ϵ.

Though pattern matching over uncertain graphs is NP-hard, a filtering-and-verification-based framework can be employed to speed up the search process [19]. It has been further proven that the pattern-matching query can be answered in polynomial time $\#P$-complete [20].

A significant amount of work has been done on the pattern-matching problem via subgraph isomorphism in deterministic graphs as well [12, 51]. The goal here is to produce a candidate answer set that is close to the exact answer set and perform a series of subgraph isomorphism tests to ensure its validity.

The graph isomorphism tests are quite expensive; in order to reduce the number of isomorphism tests, Fan et al. [19] relaxed strict query conditions by limiting the number of hops in graph patterns or integrating regular expressions as edge constraints [18]. Ma et al. [38] propose topology-constrained graph pattern matching queries, which can be processed within polynomial time [18, 19]. Additionally, Fan et al. [19] develop strategies for graph updates to avoid rerunning query algorithms. Despite these advancements, these methods primarily involve online query processing, which does not scale well for very large graphs.

In response, Cheng et al. [41] and Zou et al. [56] have explored pattern matching over large directed graphs by imposing reachability constraints. Both studies leverage pre-stored decomposed graphs in relational databases, utilizing advanced join algorithms to handle pattern-matching queries efficiently [41, 56]. These improvements emphasize a trend toward optimizing graph pattern matching to manage larger datasets effectively.

2.2.3 Influence Maximization

Influence Maximization (IM) is a critical algorithmic problem in social influence analysis that aims to select a set of k users (called the *seed set*) from a social network to maximize the expected number of users who become influenced (referred to as *influence spread*). This issue has gathered extensive attention over the past decade due to its significant potential for application and substantial technical challenges. In the realm of e-commerce, when a user purchases a product, they are said to be "influenced" or "activated". The traditional viral marketing challenge centers on identifying the top-k seed users within a

network to maximize the expected influence spread using a specific influence propagation model. The choice of k, the number of seed users, typically depends on the marketer's strategy and indicates how many consumers the marketer can initially reach and influence through methods such as advertising, free samples, and discounts.

Domingos et al. [15] formulate IM as an optimization problem modelling the network as a Markov Random field where each customer's probability of buying an object is influenced both by the desirability of the object and the influence of the other customers. Several following works have proposed new methods to improve the efficiency and accuracy of the measure. For instance, [9] developed an almost linear time algorithm for the influence maximization problem, obtaining the near-optimal approximation factor of $(1 - 1/e - \epsilon)$, for any $epsilon > 0$, in time $O((m + n)klog(n)/\epsilon^2)$, where m, n is the number of edges and vertices in the graph respectively, k is the seed size, and e is the base of the natural logarithm, while ensuring the same approximation guarantee as [15].

Numerous variations of the influence maximization problem have been explored. For instance, Barbieri et al. [8] develop a model that captures the viral adoption process of products influenced by social connections and product characteristics. They employ an iterative scaling technique to fine-tune the parameters that enhance the likelihood of successful propagation under their model. Similarly, Bharathi et al. [7] investigate the dynamics of innovation diffusion in a competitive environment where multiple firms deploy viral marketing strategies for competing products, aiming to optimize the spread of their innovations. This study uniquely considers scenarios where subsequent campaigns are aware of their competitors' initial seed nodes, adding a strategic layer to the marketing efforts. Furthermore, Lappas et al. [32] formulate the k-effectors problem, examining its complexity across various graph structures. They focus on identifying a set of k active nodes that most accurately explain the observed activation patterns within a network, according to a specific information-propagation model. This approach aims to delineate a clear activation pattern, shedding light on how information spreads through different network topologies and contributing to a deeper understanding of the mechanisms driving influence in social networks.

2.3 Streaming Graphs

This section includes definitions and notations used to describe the data processing approaches for streaming graphs. The definitions are taken from [40] and [44].

Definition 8 (Streaming Record). *A streaming record (sr) is a pair (τ, P) where τ is the event (application) timestamp of the record assigned by the data source, and P defines the payload of the record.*

Definition 9 (Streaming Graph Edge). *A streaming graph edge (sge) is a quadruple (τ, src, trg, l) where (src,trg) $\in V \times V$ are vertex pairs, l represents the label of the sge, and τ is the event (application) timestamp assigned by the external data source.*[1]

Definition 10 (Input Graph Stream). *An input graph stream is a continuously growing sequence of streaming graph edges $S^I = [sge_1, sge_2, \dots]$ where each $sge_i = (\tau_i, src_i, trg_i, l_i)$ represents an edge $e \in E$ labelled $l_i \in \Sigma$ between vertices $src_i, trg_i \in V$ and $sges$ are non-decreasingly ordered by their timestamps.*

As outlined in Chapter 1, the sheer volume of incoming data surpasses manageable processing capabilities. Therefore, it becomes crucial to establish the notion of *windows* that enables processing this unbounded data in finite batches, making it manageable. Further processing the entire stream at once would require unlimited memory and computational resources. Windowing constrains resource usage by limiting the scope of data being processed at any one time.

Definition 11 (Window). *A window of size w indexed by k, W_k, over a streaming graph is a finite multi-set of $sgrs$ denoted as a range $[W_k, W_k + w)$. Here, W_k thus represents the start time of the window.*

Definition 12 (Time-based Sliding Window). *A time-based sliding window with window size w and slide parameter β is a window that slides after every β time unit. Thus, at time unit t, the active window has index $k = \lfloor t/\beta \rfloor$ and $W_k = k \cdot \beta$. Thus, the window is given by $[W_k, W_k + w)$*

Definition 13 (Graph Snapshot). *A graph snapshot is a pair of vertex and edge sets $G = (V, E)$ forming a graph at a time point t by the $sgrs$ within a corresponding window. For simplicity, a graph snapshot is denoted as $G_k = (V_k, E_k)$ where $k = \lfloor t/\beta \rfloor$ and β is the slide parameter.*

2.3.1 Unboundedness and Real-Time Processing

As elaborated in Chapter 1, streaming data is characterized by unboundedness – i.e., it is not possible to estimate its total size. Such data is typically analyzed using stream

[1] It is assumed that a single or multiple external data source generates sges and arrive in order.

processing techniques. To frame our understanding, we draw upon insights from prior studies [40, 44] to portray streaming data through the lens where edges are integrated with time stamps.

Further, building on the groundwork established by the foundational studies [13, 48, 46, 43], this thesis delves into operations/queries on streaming graphs that require real-time processing. This aspect distinguishes it from the conventional static graph framework, wherein the graph remains unchanged and is entirely retrievable for analysis. In stark contrast, streaming graphs are in a state of constant flux, necessitating the execution of updates even when the full graph context is not present.

2.3.2 Graph Mining Problems

Numerous studies have introduced effective approaches tailored to specific graph mining tasks within the context of streaming graphs. Below, we examine some of the most extensively researched applications in the domain of streaming graph mining:

1. **Motif Counting**
   A motif is defined as any connected and unlabeled graph pattern. The process of counting motifs with a given number of vertices, known as k-motifs, involves determining the prevalence of these motifs within a graph. This task has become particularly significant in analyzing streaming graphs, where traditional methods may be inadequate due to the continuous influx of data. Research in this area often leverages probabilistic and sampling techniques to estimate motif frequencies, aiming for approximations that balance accuracy and computational feasibility. For instance, Wang et al. [49] have developed methods to sample edges from streaming data effectively, thus enabling real-time motif counting. Similarly, Chen and Lui [11] have introduced a universal unbiased estimator that adapts across various scenarios to provide reliable motif statistics, reflecting a shift towards more dynamic and adaptable analytical tools in graph processing.

2. **Pattern Matching**
   The domain of graph pattern matching has been a fertile ground for research, particularly with challenges such as the subgraph isomorphism problem, which is known to be NP-complete [21]. To address this complexity, there have been several proposals to simplify the problem into more manageable forms. For example, Fan et al. [19] introduce a matching-based bounded simulation, effectively reducing certain graph

pattern matching tasks to cubic time complexity, a significant reduction from potentially intractable scenarios. This approach was later refined to achieve quadratic time complexity [25], further optimizing the process. Additionally, ongoing innovations like those by Collins and Smith [14] focus on incremental subgraph pattern matching algorithms, which adapt to changes in the graph over time, thus underscoring the evolving nature of graph analysis tools that aim to handle dynamic data environments efficiently.

3. **Frequent Subgraph Mining**
   Frequent subgraph mining aims to identify all labelled patterns within a graph G that occur more frequently than a specified threshold τ. This challenge has traditionally been tackled by analyzing static snapshots of streaming graphs at intervals. However, recent advancements have shifted towards more continuous and dynamic methodologies, such as window-based computation, which allow for the ongoing assessment of subgraph frequencies. For instance, Abdelhamid et al. [1] have developed an incremental mining system that maintains a specialized data structure to track emerging patterns and those close to meeting frequency thresholds. This method ensures that data analysts can keep pace with rapid changes within the graph. Furthermore, techniques like those introduced by Aslay et al. [5], which employ reservoir sampling, are instrumental in detecting frequently occurring k-vertex subgraphs in real-time, illustrating a significant shift towards more responsive and timely graph analytics.

Addressing graph mining challenges in streaming graphs presents considerable difficulties primarily due to the immense sizes of resulting graphs, which often exceed the capacity of standard memory systems. Different strategies have been developed to manage these issues, focusing on both in-memory and disk-resident solutions.

For scenarios where maintaining the entire graph in memory is not feasible, some researchers have concentrated on enhancing disk-resident data management. This involves optimizing I/O access patterns to efficiently count the exact number of graph patterns without requiring extensive in-memory operations [14]. These approaches help manage large-scale data by relying on the hard disk for storage while minimizing the latency and overhead typically associated with disk access.

Conversely, other studies [1] have explored the use of in-memory algorithms that incorporate random sampling techniques. These methods selectively store portions of the induced graph in the main memory, enabling the accurate estimation of measurements

while ensuring the graph remains manageable within the available memory constraints. This strategy leverages the speed of in-memory processing while mitigating the limitations posed by memory size through intelligent sampling.

A prevalent technique for handling the unbounded nature of data streams is windowing, a cornerstone concept in stream processing. Windowing divides the continuous stream into finite segments called windows. This segmentation is crucial for performing operations like window aggregations or joins. The specific type of window used—determined by how the stream is split and which data tuples are included—greatly influences the content of the window, the outcome of windowed operations, and the feasibility of various application scenarios.

In recent years, the variety and complexity of window types have expanded significantly as researchers and practitioners seek to cater to diverse application needs. Standard fixed-size windows, such as tumbling or sliding windows, may not meet all the demands of intricate stream processing applications. Consequently, more adaptive window types have been proposed. These include data-driven windows that adjust dynamically based on the characteristics of the data, such as punctuation-based windows and session windows, which allow for more flexibility and relevance in data processing. Additionally, concepts like Frames have been introduced to tailor windowing mechanisms more closely to specific processing requirements, illustrating a broader trend toward customization in stream processing technologies. A recent article by Verwiebe1 et al. [47] gives an extensive survey of window types for stream processing systems.

Overall, these advancements reflect a growing sophistication in handling streaming graph data, emphasizing efficiency, scalability, and adaptability to a range of processing scenarios.

Chapter 3

Edge Uncertain Streaming Graphs

As discussed in the previous chapters, efficiently counting motifs in large graphs is an essential building block for analyzing the structure of large networks. One of the fundamental yet crucial substructures within these graphs is the triangle, constituting the smallest complete subgraph. Various graph metrics, including local clustering coefficients and transitivity ratios, rely on the computation of the number of triangles. Notably conspicuous in social networks, where connections often proliferate among friends of friends, triangles play a pivotal role. This ubiquitous presence across diverse networks has spurred the emergence of pivotal analytical concepts such as transitivity ratios and clustering coefficients in the domain of complex network analysis.

This chapter presents the Edge Uncertain Streaming Graph Model, aimed at approximating the count of triangles in a streaming graph. We also demonstrate the asymptotic tightness of this estimate. To achieve this, we break down the problem into two subtasks:

1. Estimating the number of active vertices: The initial phase involves accurately gauging the number of active vertices within the graph snapshot.
2. Estimating the number of triangles: Subsequent to determining the active vertices, we focus on estimating the number of triangles.

The subsequent sections of the chapter are structured as follows: Section 3.1 outlines all necessary definitions for the edge uncertain graph model, with subsequent sections focusing on each of the above subtasks individually.

3.1 Edge Uncertain Streaming Graph Model

This section combines the uncertain graph model discussed in Section 2.1 with the streaming graph model in Section 2.2 to establish and define the edge uncertain streaming graph model. This is the basis of the proposed approach to triangle counting.

Definition 14 (Edge uncertain streaming graph edge)**.** *An edge uncertain streaming graph edge is a quadruple* (τ, src, trg, p) *where* $(src, trg) \in V \times V$ *is an edge between vertices* $src, trg \in V$*,* $p \in [0, 1]$ *denotes the probability of the existence of this edge, and* τ *is the event timestamp.*

Definition 15 (Edge uncertain graph snapshot)**.** *An edge uncertain graph snapshot (EUG) over a window* W_k *is a triple* $\mathcal{G}_{W_k} = (V_{W_k}, E_{W_k}, p_{W_k})$ *where* V_{W_k} *and* E_{W_k} *are vertices and edges that are in window* W_k *and* p_{W_k} *is the set of probabilities of* E_{W_k}*.*

In the remainder, we omit the window subscript for simplicity and simply refer to the snapshot as $\mathcal{G}_k$.

3.2 Estimating the Number of Active Vertices

As previously mentioned, a crucial step in estimating the triangle count within a window involves first determining the number of active vertices it contains. Essentially, the goal is to forecast the count of active vertices in an upcoming window without having to list each one explicitly. To achieve this, we employ a probabilistic model that provides predictions. We demonstrate that this model is both unbiased and precise in its estimations.

Consider a *EUG* snapshot $\mathcal{G}_k$ at a given time t. Suppose the count of vertices in the preceding *EUG* snapshot $\mathcal{G}_{k-1}$ is denoted as n. For each incoming edge $e_t = (t, src, trg, p_e)$ in the graph, the following probabilities are discerned:

- The probability that both vertices src and trg are already included in $\mathcal{G}_{k-1}$ equals $\frac{1}{n^2}$. Consequently, the vertex count in $\mathcal{G}_k$ remains n.
- The probability that only one of the vertices src or trg was present in $\mathcal{G}_{k-1}$ is $2p_e\left(1 - \frac{1}{n-1}\right)\frac{1}{n}$, This indicates that the vertex count in $\mathcal{G}_k$ increases to $n + 1$.
- The probability that neither vertex src nor vertex trg was present in $\mathcal{G}_{k-1}$ is given by $p_e\left(1 - \frac{1}{n}\right)^2$, resulting in an increase to $n + 2$ for the vertex count in $\mathcal{G}_k$.

We define a random variable X_e that represents the number of vertices incident to an edge $e \in E_k \setminus E_{k-1}$ added to $\mathcal{G}_{k-1}$ at time t. Consequently, the total number of vertices added due to the shift in this time window can be given by

$$n' = \sum_{e \in E_k \setminus E_{k-1}} X_e$$

The expected value of n' is derived using:

$$\mathbb{E}[n'] = \mathbb{E}\left[\sum_{e \in E_k \setminus E_{k-1}} X_e\right]$$

and by the linearity of expectations, we can write -

$$\mathbb{E}[n'] = \sum_{e \in E_k \setminus E_{k-1}} \mathbb{E}[X_e].$$

Now, considering all possible probabilities as discussed above, we can obtain $\mathbb{E}[X_e]$ by doing the following calculations.

$$\begin{aligned}\mathbb{E}[X_e] &= 0 \cdot \frac{1}{n^2} + 1 \cdot 2p_e \left(1 - \frac{1}{n-1}\right)\frac{1}{n} + 2 \cdot p_e \left(1 - \frac{1}{n}\right)^2 \\ &\approx 2p_e \left(1 - \frac{1}{n}\right)\end{aligned}$$

Intuitively, this means that for every new edge $e \in E_k \setminus E_{k-1}$ the expected number of new vertices it adds to the window k is approximately $2p_e\left(1 - \frac{1}{n}\right)$. Enumerating this expectation for all edges in the set $E_k \setminus E_{k-1}$, by linearity of expectation, we can say -

$$\mathbb{E}[n'] \approx 2\left(1 - \frac{1}{n}\right) \sum_{e \in E_k \setminus E_{k-1}} p_e \tag{3.1}$$

Here, $E[n']$ is the expected number of vertices added to window $k-1$ when it slides from $k-1$ to k.

It should be emphasized that our analysis thus far has not considered the potential impact of edge deletions on the estimate. Under the sliding window model, some vertices present in $\mathcal{G}_{k-1}$ may no longer be relevant in $\mathcal{G}_k$ because they "expire" or become obsolete

when the window slides. To illustrate this concept, refer to the example depicted in Figure 3.1. In this figure, we observe a streaming graph in which edges are introduced over time. For instance, at time t_0, we witness the arrival of the edge (A, B), followed by the edge (B, C) at t_3, and so forth.

For this example, let us assume a window size of four units and a slide interval of two units. The first window, i.e., the window corresponding to $k = 1$, encompasses all activity from time $t = t_0$ to $t = t_4$, resulting in a graph snapshot as shown in Figure 3.2. As the window advances, we migrate from $\mathcal{G}_1$ to $\mathcal{G}_2$ which corresponds to the time interval $[t_2, t_6]$. The subsequent snapshot, as visualized in Figure 3.3, does not include the edge (A, B) because it falls outside of $\mathcal{G}_2$'s temporal scope. In this scenario, we classify the edge (A, B) as having "expired". Moreover, as the window slides again to $k = 3$ corresponding to the interval $[t_4, t_8]$, we further observe the expiration of the edge (B, C), which consequently renders the vertex B defunct within that snapshot. Thus, it becomes important to accommodate this phenomenon of vertex "expiry" while estimating the count of active vertices, ensuring our model accurately reflects the fluid nature of the graph's topology over time.

Furthermore, the disappearance of a vertex from one window does not necessarily imply its permanent removal from the entire graph sequence. As indicated in Figure 3.5 vertex B again makes a reappearance in the snapshot $\mathcal{G}_4$, demonstrating the recurrent nature of vertices in the graph.

To incorporate the concept of vertex expiration, as previously outlined, we reformulate our methodology using a probabilistic strategy. It is crucial to understand that within the context of our streaming model, it is the edges that dictate the inclusion and exclusion of vertices in a graph snapshot; not the other way around. Therefore, all our incremental computation is centred toward the arrival and deletion of edges, not vertices.

When the window slides from $k-1$ to k, for a vertex associated with an expiring edge, i.e., $e \in E_{k-1} \setminus E_k$, its inclusion in V_k is only possible if its degree at time t is ≥ 1. If this condition is not met, the vertex is excluded from V_k and thus is not present in the EUG snapshot $\mathcal{G}_k$

More formally, let $d(u)$ represent the total count of edges connected to u $\forall u \in V_k$. For every vertex u associated with $e \in E_{k-1} \setminus E_k$, if $d(u) \geq 1$, it is still present in $\mathcal{G}_k$ when the window moves from $k-1$ to k. By definition, for deterministic graphs, $d(u) = (\#$ of edges incident to $u)$. For uncertain graphs, however, $d(u)$ is also a probabilistic quantity, which entirely depends on the probability of the existence of edge incident to u. To calculate this, let us introduce a binary random variable $Y_e \in \{0, 1\}$ corresponding to an edge e such that $Y_e = 1$ if e exists in the PWG, and 0 otherwise. It can be easily said that $Pr(Y_e = 1) = p_e$

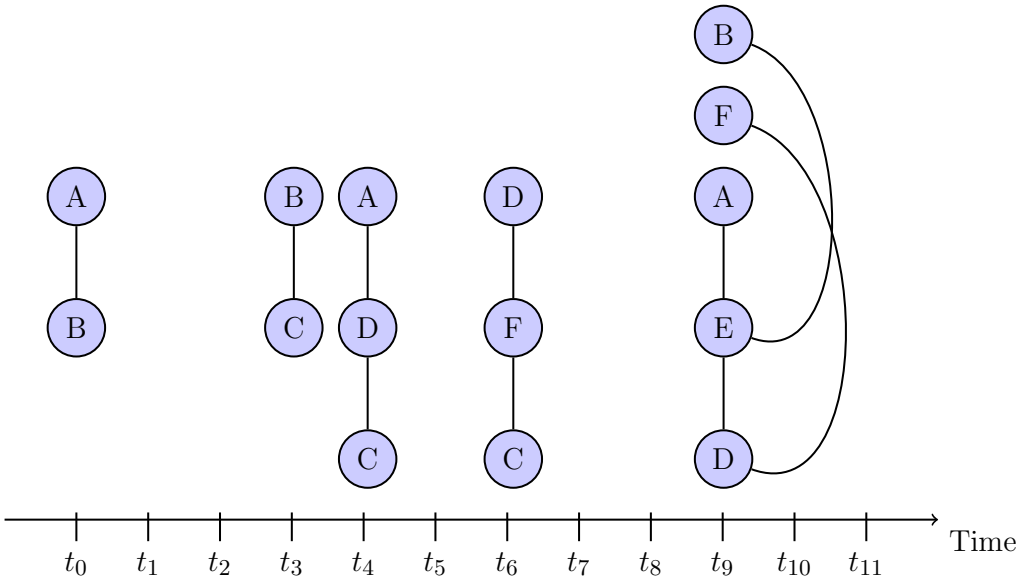


Figure 3.1: Streaming graph

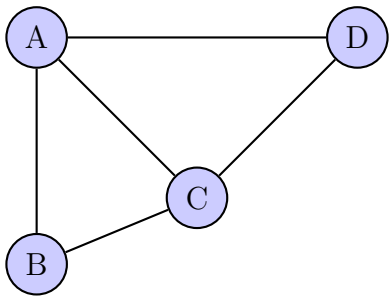


Figure 3.2: Graph Snapshot $\mathcal{G}_1$ corresponding to $[t_0, t_4]$

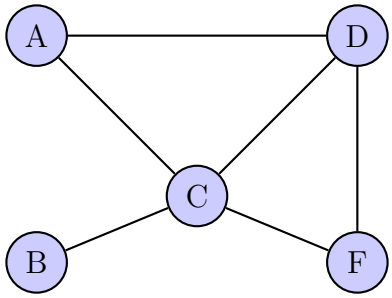


Figure 3.3: Graph Snapshot $\mathcal{G}_2$ corresponding to $[t_2, t_6]$

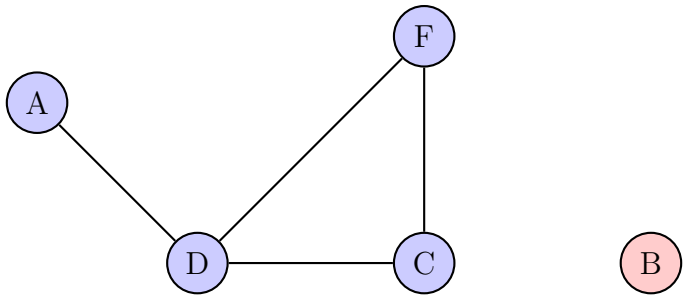


Figure 3.4: Graph Snapshot $\mathcal{G}_3$ corresponding to $[t_4, t_8]$. Notice that the node B has expired.

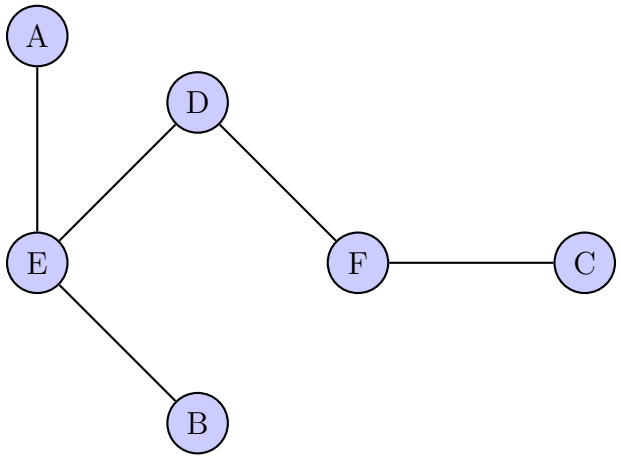


Figure 3.5: Graph Snapshot $\mathcal{G}_4$ corresponding to $[t_6, t_{10}]$

and $Pr(Y_e = 0) = 1 - p_e$. Further, for any graph $G \in PWG(\mathcal{G}_k)$,

$$d(u) = \sum_{e \in I_u} Y_e$$

where I_u is the set of edges incident to u in $\mathcal{G}_k$.

And for u in order to be relevant in the next window, it should have $d(u) \geq 1$. The probability of this event can be given by

$$Pr[d(u) \geq 1] = Pr\left[\sum_{e \in I_u} Y_e \geq 1\right]$$

applying Markov's inequality to the above equation, we get

$$\begin{aligned} Pr[d(u) \geq 1] = Pr\left[\sum_{e \in I_u} Y_e \geq 1\right] &\leq \frac{E\left[\sum_{e \in I_u} Y_e\right]}{/} 1 \\ &= \sum_{e \in I_u} p_e \end{aligned}$$

This result essentially says - that for a vertex u associated with an edge that expires when the window moves, there is a probability $\leq \sum_{e \in I_u} p_e$ that it still sustains in the next window. To make the analysis easier, let us consider another random variable $X_v \in \{0, 1\}$ for every vertex $v \in V_{k-1}$ such that $X_v = 0$ if the vertex v is removed from the snapshot when the window slides, and 1 otherwise. i.e.

$$X_v = \begin{cases} 1 & \text{if } d(v) \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

From the above arguments, it's easy to note that

$$Pr[X_v = 1] = Pr[d(u) \geq 1] \leq \sum_{e \in I_u} p_e$$

Note that this argument is for a single vertex. Extrapolating it to encompass all vertices linked to edges that are set to expire, we can express the count of vertices, denoted as n'' that are not removed from $\mathcal{G}_k$ when the window slides can be given by

$$n'' = \sum_{e \in E_{k-1} \setminus E_k} \sum_{u \in S} X_u$$

where $S = (src, trg)$ associated to the edge e.

Using linearity of expectations, the expected value of n'' can be derived as

$$\mathbb{E}[n''] = \sum_{e \in E_{k-1} \setminus E_k} \sum_{u \in S} \mathbb{E}[X_u] \tag{3.2}$$

$$= \sum_{e \in E_{k-1} \setminus E_k} \sum_{u \in S} Pr[d(u) \geq 1] \tag{3.3}$$

$$\leq \sum_{e \in E_{k-1} \setminus E_k} \sum_{u \in S} \sum_{e \in I_u} p_e \tag{3.4}$$

$$= 2 \sum_{e \in E_{k-1} \setminus E_k} p_e \tag{3.5}$$

Combining Equation 3.1 and Equation 3.5, the total number of vertices $|V_t|$ in the window corresponding to time t is upper bounded by

$$|V_k| \leq 2\left(1 - \frac{1}{n}\right) \sum_{e \in E_t \setminus E_{k-1}} p_e + 2 \sum_{e \in E_{k-1} \setminus E_k} p_e + \left| \bigcup_{e \in E_{k-1} \setminus E_k} (src_e, trg_e) \right| \tag{3.6}$$

Where the last term corresponds to the vertices associated with the edges that do not expire.

3.3 Active vertex estimation example

In this section, we detail the process of estimating the active vertex count. We exemplify this process using a simple example EUG snapshot $\mathcal{G}_k$ at time t as shown in Figure 3.6. At time $t+1$ assume two new *sge*'s $(t+1, D, F, 0.9)$ and $(t+1, I, J, 0.2)$ appears and one existing edge $(B, C, 0.2)$ expires. By the estimate proposed above, the edges corresponding to $E_t \setminus E_{k-1}$ are the new edges that come in at time $t+1$, i.e., (D, F) and (I, J). The edge that is expiring is (B, C), and for our case, there are 5 vertices $\{A, B, C, D, E\}$, which are associated with edges that do not expire when the window slides.

By our estimate, the total number of new vertices in the window corresponding to $t+1$

can be calculated by

$$
\begin{aligned}
|V_{t+1}| &\leq 2\left(1-\frac{1}{n}\right)\sum_{e\in E_k\setminus E_{k-1}} p_e + 2\sum_{e\sum_{e\in E_{k-1}\setminus E_k}} p_e\left|\bigcup_{e\in E_{k-1}\setminus E_k}(src_e, trg_e)\right| \\
&= 2\left(1-\frac{1}{5}\right)(0.9+0.2)+2(0.2)+5 \\
&= 7.16
\end{aligned}
$$

As shown in the figure, this estimate is very close (slightly larger) to what we observe. Further, by the *law of large numbers*, we can say that the estimate becomes even closer when we have a large number of vertices in the window.

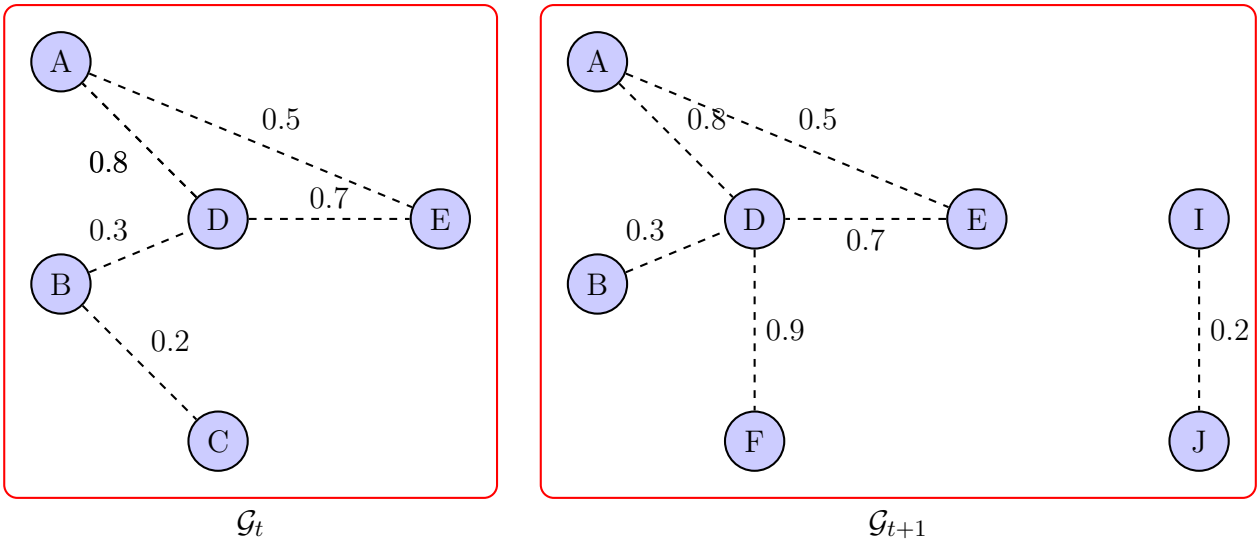


Figure 3.6: Example Graph

3.4 Estimating the Number of Triangles

To estimate the number of triangles in an EUG snapshot G_k, we model it as **edge exposure martingales** [4]. Martingales are a sequence of random variables that describe a stochastic process where, at a particular time, the conditional expectation of the next value in the sequence is equal to the present value, regardless of all prior values.

Definition 16 (Martingale)**.** *A sequence of random variables $X_1, X_2, \ldots,$ is said to be a martingale sequence if for all $i > 0$, we have*

$$\mathbb{E}[X_i | X_1, \ldots, X_{i-1}] = X_{i-1}$$

The notion of martingales has its roots in the gambling arenas of Western Europe, where it became known through the eponymous betting system. In this system, gamblers would respond to a loss by wagering an amount that would recoup all previous losses. In the era predating the imposition of table limits, this strategy was perceived as virtually failproof, contingent on the assumption that an eventual win would nullify all accumulated losses. However, the critical question arises: what occurs when a gambler runs out of money and is unable to place a further bet?

Specifically, consider the game of roulette, where the objective is to predict the final resting sector of a small white ball on a wheel divided into 38 segments: 18 black, 18 red, and 2 green. The game's operation is simple: the wheel is spun, wagers are laid, and the ball is released onto the wheel in motion. Over time, friction diminishes the wheel's rotation, culminating in the ball settling into one of the segments.

Bettors stake their money on the colour—red or black—of the segment in which the ball will halt, with even odds being the norm. If a gambler stakes k and their prediction holds, they receive $2k$; if not, they lose their money.

Let X_i denote the gambler's wealth after i betting rounds. This amount is the previous wealth X_{i-1} adjusted by the wager's outcome. Although the outcomes of successive spins of the roulette wheel are independent events, the gambler's wagers are not – they depend on all previous outcomes of the roulette wheel (thus the gambler's current fortune). Therefore, X_i does not represent the aggregate of an independent random variable series. With a probability of 18/38, the gambler anticipates a successful wager, but with a probability of 20/38, expects a loss.

Consequently, the expected value of X_{i+1}, given the historical data of the wheel and the gambler's funds, is invariably lower than X_i. Thus, the gambler is perpetually poised to lose on the next bet. At some point T, the betting ceases, and the focal point becomes the expected value of the gambler's wins/losses at this point. One can appeal to the theoretical underpinnings of martingales to ascertain this expected value. As will be elaborated in subsequent sections, probabilistic techniques render the calculation of this expectation not only straightforward but also highly accurate.

Martingale's theories extend their influence beyond the realms of finance and gambling, penetrating varied sectors like public health and computational intelligence. For instance,

in public health, martingale methods are applied to the modelling of epidemics [50], providing a structure for forecasting the trajectory and zenith of disease transmission amid uncertainty. In the domain of computational intelligence, martingales offer a framework for evaluating the efficacy of machine learning algorithms [13], particularly in environments characterized by the processing of unpredictable data sequences and projecting their anticipated evolution through successive rounds.

3.4.1 Doob or Exposure Martingales [4]

Consider $\Omega_1, \Omega_2, \ldots, \Omega_n$ as probability spaces defined over discrete sets $S_1, S_2, \ldots, S_n$ respectively, and an arbitrary function $f : S_1 \times \cdots \times S_n \to \mathbb{R}$. Let $\vec{y} = \{y_1, y_2, \ldots, y_n\}$ such that y_1 is sampled independently from S_1, y_2 from S_2, and so on.

For a given vector $\vec{a} \in S_1, S_2, \ldots, S_n$, we can then define a sequence of random variables $X_0, \ldots, X_n$ as follows:

$$
\begin{aligned}
X_0(\vec{a}) &= \mathop{\mathbb{E}}_{\vec{y}}[f(\vec{y})], \\
X_1(\vec{a}) &= \mathop{\mathbb{E}}_{\vec{y}}[f(\vec{y}) | y_1 = a_1], \\
X_2(\vec{a}) &= \mathop{\mathbb{E}}_{\vec{y}}[f(\vec{y}) | y_2 = a_2 \wedge y_1 = a_1], \\
&\vdots \\
X_i(\vec{a}) &= \mathop{\mathbb{E}}_{\vec{y}}[f(\vec{y}) | \bigwedge_{j \leq i} (y_j = a_j)]
\end{aligned}
$$

where the subscript in $\mathbb{E}_{\vec{y}}$ indicates that the expectation is taken over the random choice of $\vec{y}$. It is crucial to note that $X_n(\vec{a}) = f(a_1, a_2, \ldots, a_n)$

Lemma 3.4.1. *The sequence of random variables $X_0, \ldots, X_n$ is a martingale.*

Proof. We need to verify that for arbitrary values of $x_0, \ldots, x_i$ we have:

$$\mathbb{E}[X_{i+1} | (X_i = x_i) \wedge \cdots \wedge (X_0 = x_0)] = x_i$$

Let E be the event $(X_i = x_i) \wedge \cdots \wedge (X_0 = x_0)$. Using the definition of expectation, we can say:

$$\mathop{\mathbb{E}}_{\vec{y}}[X_{i+1}(\vec{y})|E] = \sum_{\vec{a}\in(S_1,S_2,\ldots,S_n)} \mathbb{E}\left[X_{i+1}|E \wedge \bigwedge_{j\le i} y_j = a_j\right] Pr\left[\bigwedge_{j\le i}(y_j = a_j)|E\right]$$

Notice that the summands where $\vec{a} \notin E$ would cancel, and we'll be left with

$$\sum_{\vec{a}\in E} \mathbb{E}\left[X_{i+1}|E \wedge \bigwedge_{j\le i} y_j = a_j\right] Pr\left[\bigwedge_{j\le i}(y_j = a_j)|E\right]$$

and it suffices to verify that, whenever $X_i(\vec{a}) = x_i$

$$\mathop{\mathbb{E}}_{\vec{y}}\left[X_{i+1}(\vec{y})|\bigwedge_{j\le i}(y_j = a_j)\right] = x_i$$

Let Pr_i denote the probability on Ω_i. As $X_{i+1}(\vec{y})$ only depends on $(y_1, y_2, \ldots, y_n)$, we can say:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\vec{y}}\left[X_{i+1}(\vec{y})|\bigwedge_{j\le i}(y_j = a_j)\right] &= \sum_{y_{i+1}\in S_{i+1}} Pr_{i+1}[y_{i+1}]X_{i+1}(a_1, a_2, \ldots, a_n, y_{i+1}) \\
&= \sum_{y_{i+1}\in S_{i+1}} Pr_{i+1}[y_{i+1}]\mathop{\mathbb{E}}_{\vec{b}}[f(\vec{b})|(b_{i+1} = y_{i+1} \wedge \bigwedge_{j\le i}(b_j = a_j)] \\
&= \mathop{\mathbb{E}}_{\vec{b}}[f(\vec{b})|\bigwedge_{j\le i}(b_j = a_j)] \\
&= X_i(\vec{a})) = x_i
\end{aligned}$$

Here the sequence $X_0, \ldots, X_n$ is the *exposure martingale* for the function f. □

Let us try to understand this with an example. An exposure martingale can be understood through the lens of a repetitive betting game that unfolds in real time. Imagine that each round of the game, labelled as trial k, yields a potential reward X_k. The cumulative experience of the game up to the i^{th} trial is represented by the sequence $X_1, \ldots, X_{i-1}$. The stake for the i^{th} trial is strategically determined based on the results of all prior trials; a history of wins may encourage the player to increase their bet, whereas a sequence of

losses might prompt a more conservative wager. Consequently, the earnings from the current trial are contingent on the invested amount, which is directly linked to the outcomes of all preceding trials.

This interdependence means that as the game progresses, the outcomes from each trial progressively "unveil" or "expose" the trajectory of the game, reflecting the core principle of an exposure martingale. In such a scenario, the current state of play provides no advantage in predicting the outcome of the subsequent trial beyond what was already known. Each trial's result serves as a revelation that informs the next decision, creating a chain of interconnected outcomes that epitomize the unfolding of an exposure martingale in a real-world setting.

Given this motivation, we look at some strong probabilistic results that strengthen the martingale model even more. Specifically, we look at concentration inequalities related to this exposure model that would help us verify our calculated estimates with utmost certainty.

Theorem 3.4.2 (Azuma's Inequality [4])**.** *Let $c = X_0, \ldots, X_n$ be an exposure martingale with the property*

$$|X_{i+1} - X_i| \leq 1$$

for all $0 \leq i \leq n$. Then

$$Pr[|X_n - c| > \lambda\sqrt{n}] < 2e^{-\lambda^2/2}$$

where the condition $|X_{i+1} - X_i| \leq 1$ is called the Lipschitz Conditions.

Proof. Consider $Y_i = X_i - X_{i-1}$ so that $|Y_i| \leq 1$ and $\mathbb{E}[Y_i|X_{i-1}, \ldots, X_0] = 0$. Then for $\alpha > 0$, Set

$$h(x) = \frac{e^{\alpha} + e^{-\alpha}}{2} + \frac{e^{\alpha} - e^{-\alpha}}{2}x$$

Then $\forall x \in [-1, 1]$, $e^{\alpha x} \leq h(x)$ as $y = h(x)$ is a chord through the points $x = \pm 1$ of the convex curve $y = e^{\alpha x}$. Thus,

$$\begin{aligned}
\mathbb{E}[e^{\alpha Y_i}|X_{i-1}, \ldots, X_0] &\leq \mathbb{E}[h(Y_i)|X_{i-1}, \ldots, X_0] \\
&= h(\mathbb{E}[Y_i|X_{i-1}, \ldots, X_0]) \\
&= h(0) \\
&= \cosh \alpha \\
&\leq e^{-\alpha^2}
\end{aligned}$$

Further,

$$\begin{aligned}\mathbb{E}[e^{\alpha X_n}] &= \mathbb{E}[\prod_{i=1}^{n} e^{\alpha Y_i}] \\ &= \mathbb{E}[(\prod_{i=1}^{n-1} e^{\alpha Y_i})\, \mathbb{E}(e^{\alpha Y_m} | X_{m-1}, \ldots, X_0)] \\ &\leq \mathbb{E}[\prod_{i=1}^{n-1} e^{\alpha Y_i}] e^{-\alpha^2/2}\end{aligned}$$

Using Markov's Inequality, we can further say -

$$Pr[X_n > \lambda\sqrt{n}] = Pr[e^{\alpha X_n} > e^{\alpha\lambda\sqrt{n}}] < \mathbb{E}[e^{\alpha X_n}]/e^{\alpha\lambda\sqrt{n}} \leq e^{\alpha^2 n/2 - \alpha\lambda\sqrt{n}}$$

Setting $\alpha = \lambda/\sqrt{n}$

$$Pr[X_n > \lambda\sqrt{n}] < e^{-\lambda^2/2}$$

and further by symmetry,

$$Pr[|X_n - c| > \lambda\sqrt{n}] < 2e^{-\lambda^2/2} \tag{3.7}$$

□

The conclusion drawn above provides a significant insight into the behavior of processes modeled as martingales. Specifically, it proves that if a martingale process adheres to the condition $|X_{i+1} - X_i| \leq 1$, i.e., the absolute difference between successive terms of the process does not exceed one, then the real-world manifestation of this process will align closely with its theoretical expected value. This condition essentially implies a constraint on the volatility or variance within the process, ensuring that changes from one term to the next can be calculated with bounded error.

3.4.2 Edge Exposure Martingale

The discourse on martingales thus far has been situated within the framework of temporal processes, wherein the outcome at any given moment is a function of preceding outcomes. The challenge arises when we attempt to apply this martingale model to the domain of graph theory. We are about to delve into how this model is applicable within the context

of uncertain graphs. Let us consider a visual example to cultivate an intuitive grasp of this concept. For the sake of simplicity, we take into account an edge uncertain graph denoted as $G(n, p)$ where n represents the number of vertices and each pair of vertices has a probability p of being connected by an edge. Consequently, the total number of possible edges in this graph is $m = \binom{n}{2}$. [1]

If we consider, for a moment, the vertex colouring problem in the setting of an EUG, the problem presents itself quite naturally for visual comprehension. Our motive here is to provide an intuitive understanding of the model and then talk about the rigorous mathematical proofs. Consider the vertex colouring problem, where the objective is to determine the smallest set of unique colours needed to colour every vertex of a given graph so that no two adjacent vertices share the same colour. For illustrative purposes, we will explore a basic case where $n = 3$ and $m = 3$. In Figure 3.7, it is postulated that the edges become evident sequentially in the following order: "bottom, left, right".

Figure 3.7 demonstrates how, as each edge is revealed—or 'exposed'—in the stipulated sequence, we can apply a colouring strategy that adapts to the evolving structure of the graph. Note that there are 2^3 sample graphs, and each has a probability $\frac{1}{8}$ of being materialized. Let us go through the example step by step:

1. When we have no information about the edges, the expected number of colours for the given case can be calculated by

$$\frac{1}{8}(1 + 2 + 2 + 2 + 2 + 2 + 2 + 3) = 2 = X_0$$

2. Upon the disclosure of the initial edge, which in our scenario is the bottom edge, there are two potential outcomes to consider. This edge might be present, which occurs with a probability of 0.5, or it might be absent in the actual graph. Thus, the figure branches into two: if the bottom edge exists, we need at least two colours as there exists an edge between a pair of vertices, but if it doesn't, we can get away with using just one colour. That is,

   - if the bottom edge exists, as shown in the graph, the expected number of colours required $= \frac{1}{4}(2 + 2 + 2 + 3) = 2.25$
   - similarly, if the bottom edge is absent, as shown in the graph, the expected number of colours required $= \frac{1}{4}(2 + 2 + 2 + 1) = 1.75$

[1] This example has been taken from [4]

So X_1 can take two values depending on whether or not the bottom edge is present. Note that

$$E[X_1|X_0] = \frac{1}{2} \cdot 2.25 + \frac{1}{2} \cdot 1.75 = 2 = X_0$$

which satisfies the very definition of the martingale model.

3. Similarly, when we receive information about the left edge, the analysis further branches into two. Again note that

   - $E[X_2|X_1, X_2] = \frac{1}{2} \cdot 2.5 + \frac{1}{2} \cdot 2 = 2.25 = X_1$ for the case where the bottom edge was present
   - $E[X_2|X_1, X_2] = \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 1.5 = 1.75 = X_1$ for the case where the bottom edge was absent.

4. Same analysis can be extended for X_3 and it can be easily checked from the figure that it is consistent with the martingale model

It is interesting to note that this particular modelling of the vertex colouring problem aligns with the criteria mentioned in Theorem 3.4.2. While this can be readily verified by referencing Figure 3.7, we can also visualize it in a more straightforward way. At the step where an additional edge is exposed, in the worst case, the resolution could simply involve assigning an entirely new colour to one of the vertices connected by the newly exposed edge. This would increase the count of total distinct colours required by 1, which is consistent with Theorem 3.4.2.

For the scope of this thesis, our objective is to estimate and bound the number of triangles in an EUG snapshot $\mathcal{G}_k = (V_k, E_k, p_k)$, where $|V_k| = n$ and $|E_t| = m$. Following from the above example, we first order the edges in E_k as $e_1, e_2, \ldots, e_m$. Let $f : 2^m \to \mathbb{R}$ be a function where $f(G)$ gives the number of triangles in G and $G \in PWG(\mathcal{G}_k)$ is a graph sampled from $\mathcal{G}_k$.

We define a martingale $X_1, \ldots, X_m$ by giving the values $X_i(H)$ where $G \in PWG(\mathcal{G}_k)$:

$$X_i(H) = \mathbb{E}\left[f(G)|e_j \in G \longleftarrow e_j \in H, 1 \leq j \leq i\right] \tag{3.8}$$

Note that $X_m(H)$ is simply $f(H)$ and $X_0(H) = \mathbb{E}[f(H)]$ where H is sampled from $\mathcal{G}_k$.

In other words, to determine $X_i(H)$, the process commences with the revelation of the first i edge pairs, denoted as $e_1, e_2, \ldots, e_i$, check if they are present in H. Subsequent edges

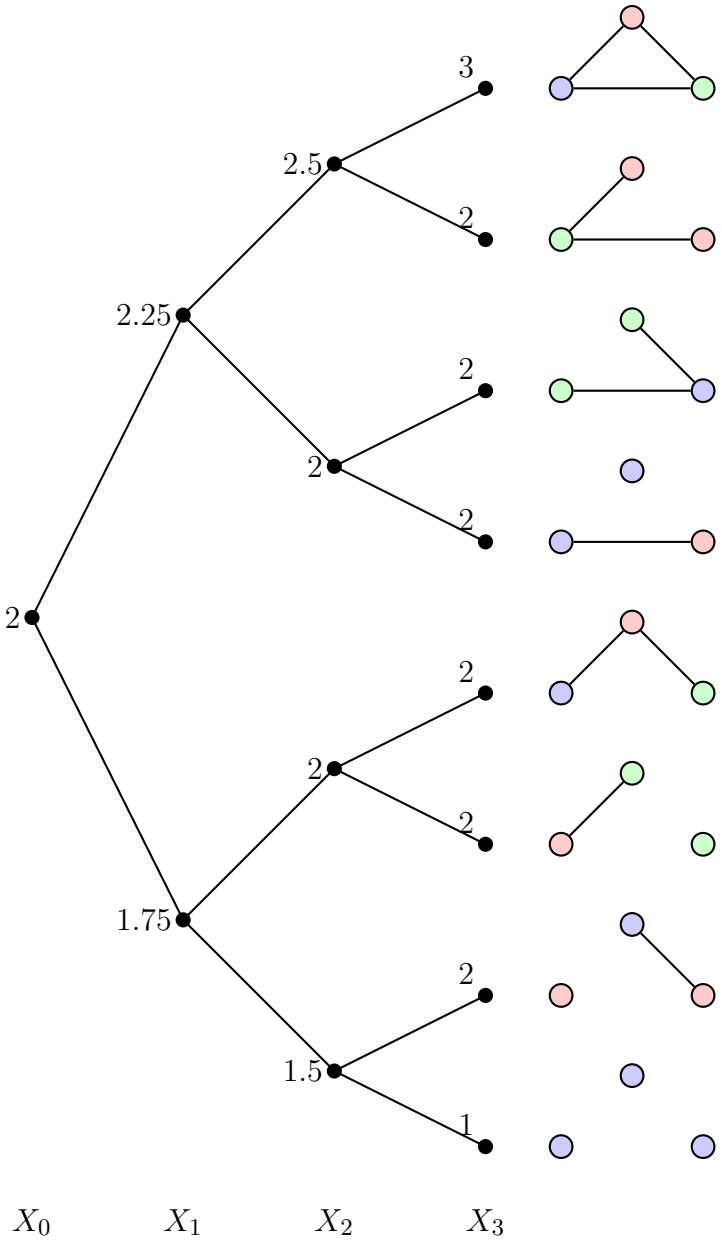


Figure 3.7: The edge exposure martingale with $n = m = 3$, representing the number of distinct colours required for acceptable vertex colouring. The edges are exposed in the order "bottom, left, right". The value of X_i are given by tracing from the central node to the leaf.

remain unseen and are presumed to follow a random distribution. In this context, $X_i(H)$ is then the conditional expectation of $f(G)$ based on this partial insight. Specifically, when $i = 0$, no edges have been disclosed, and thus, X_0 is a constant with its value equal to $\mathbb{E}[f(H)]$. On the contrary, at $i = m$, every edge is revealed, thus $X_m = f(H)$. The martingale, thus, moves from no information to full information in small steps.

Further, we observe that when a single of one edge, there can be an increase of maximum $n - 2$ triangles in the given sample. i.e.

$$|f(H) - f(H')| \leq n - 2 < n$$

Where H and H' differ by one edge. In order to have a function that satisfies the constraint discussed in Theorem 3.4.2, we define another function $g(H)$ such that $g(H) = f(H)/n$. And thus,

$$|g(H) - g(H')| \leq 1$$

Here, the function g satisfies the *Lipschitz Condition* as stated in Theorem 3.4.2. Thus, the corresponding edge exposure martingale $Y_0, \ldots, Y_m$, also satisfies

$$|Y_i - Y_{i-1}| \leq 1$$

As proved in Theorem 3.4.2 we directly get our result

$$Pr[|g(H) - c| \geq \lambda\sqrt{m}] > 2e^{-\lambda^2/2} \tag{3.9}$$

where $c = \mathbb{E}[g(H)]$ and $\mathbb{E}[g(H)]$ can be found using Algorithm 1. Note that $N(v) \forall v \in V_k$ returns the set of neighbours of v.

Further, if we make $\lambda \to \infty$ arbitrarily slowly, we can see that the distribution of $g(H)$ is tightly concentrated around c.

We employ this finding to establish the appropriate value of λ and offer an estimate for the number of triangles present in a given snapshot. Subsequently, to validate the correctness of our derivation, comprehensive experiments are presented and analyzed in Section 3.6.

It's crucial to emphasize that this approach is applicable only to motifs conforming to the Lipschitz condition when represented as Martingales. For subgraphs with a huge number of edges, the count of subgraphs completed upon adding a specific edge is $\geq O(n)$, thereby breaching asymptotic tightness when transitioning from function f to g.

Algorithm 1: Calculating expected number of triangles in an EUG snapshot

Input : EUG Snapshot $\mathcal{G}_{\|} = (V_k, E_k, p_k)$
Output: $E[g(H)]$ where $H \in PWG(\mathcal{G}_k)$

$E[g(H)] = 0$ **for** $v_0 \in V_k$ **do**
  for $v_1 \in N(v_0)$ **do**
    for $v_2 \in N(v_1)$ **do**
      $E[g(H)] = E[g(H)] + p_{v_0,v_1} p_{v_1,v_2} p_{v_2,v_0}$
    end
  end
end

3.5 Triangle count estimation example

Consider the same example as given in Figure 3.6. Using the algorithm discussed above, the number of triangles in $\mathcal{G}_k$ can be calculated by

$$p_{(A,D)} p_{(D,E)} p_{(E,D)} = 0.28$$

which essentially says that out of 100 possible world graphs generated from $\mathcal{G}_k$, 28 of them would have just 1 triangle.

3.6 Experimental Evaluation

In this section, we present the experimental evaluation that utilizes synthetically constructed streaming graphs, which allows for the precise management of vertex degree distribution and verification that the input streaming graph adheres to our presupposed characteristics. First, we analyze various real-world graphs derived from open-source repositories such as SNAP [34] and Konect [2]. Leveraging insights from these analyses, we synthesize a graph that replicates these real-world graph characteristics. Subsequently, we imbue this synthetic graph with random probability values and timestamps corresponding to each edge in the graph to resemble a real streaming scenario.

To simulate a streaming environment reminiscent of live data flow, we use Apache Kafka. This setup involves a producer component that dispatches streaming data, com-

[2]Konect - http://konect.cc/

prising edges as previously described, and a consumer component tasked with ingesting this edge data to give an estimated count of triangles. For the purpose of validation, within each window of analysis, we draw a sample graph based on the assigned edge probabilities and execute a brute-force triangle count on this sampled graph. This empirical count is then set against our estimated values to gauge accuracy.

An extensive exposition of each component within this experimental framework, encompassing the generation of synthetic graphs, the streaming simulation through Apache Kafka, and the validation methodology, will be provided in the subsequent subsections.

3.6.1 Data Generation

We employ a synthetic data generator designed to replicate the attributes of real-world streaming graphs closely. We use the NetworkX library [22] from SciPy to emulate the preferential attachment graph. This simulation is pivotal for our research, as it allows us to test our hypotheses and methodologies in a controlled environment that reflects realistic conditions. To establish this realistic simulation, our approach incorporates the analysis of various real-world network graphs. We specifically examine the degree distribution of each vertex within these graphs, a critical step that involves calculating the probability mass function (PMF) for a given degree k, denoted as PMF(k). This function represents the likelihood that a given vertex within the network has a degree of k.

An important aspect of our analysis involves the graphical representation of PMF(k) against k on a $\log - \log$ scale. Given that triangle counting is crucial for the calculation of clustering coefficients— a metric pivotal in the study of network structure — we selectively target datasets that exhibit either a) high or b) low clustering coefficients. Building upon this selection criterion, we proceed to craft synthetic datasets that closely mirror these specific clustering attributes.

Figure 3.8 illustrates the degree distribution within an undirected network sourced from the KONECT project. This network charts the interactions among free-ranging Eastern Grey Kangaroos (*Macropus giganteus*) within the Nadgee Nature Reserve located in New South Wales, Australia. Each vertex corresponds to an individual kangaroo, and edges indicate recorded interactions. The graph is characterized by a high clustering coefficient, indicative of the tendency for certain kangaroo groups to interact closely. Conversely, Figure 3.11, derived from the SNAP repository, delineates the friendship relationships in a Facebook dataset. This network is distinguished by a notably lower clustering coefficient.

Despite this disparity in clustering coefficients, the degree distribution curves for both the kangaroo and Facebook networks exhibit a marked similarity.

As shown in Figure 3.10 and Figure 3.11, notably, for higher values of k, these plots demonstrate a linear relationship, suggesting that the degree distributions of these real-world networks exhibit scale-free properties. The data generator (taking ideas from [44]) adopts a preferential attachment process to replicate these scale-free characteristics in the synthetic data. This is operationalized through the utilization of the well-known Barabási–Albert (BA) model, a method for generating random networks that inherently exhibit scale-free properties. By leveraging this model, as shown in Figure 3.13, we are able to generate synthetic networks that not only bear resemblance to real-world data in terms of structure but also in how nodes within these networks are interconnected, thus providing a robust foundation for our subsequent analyses.

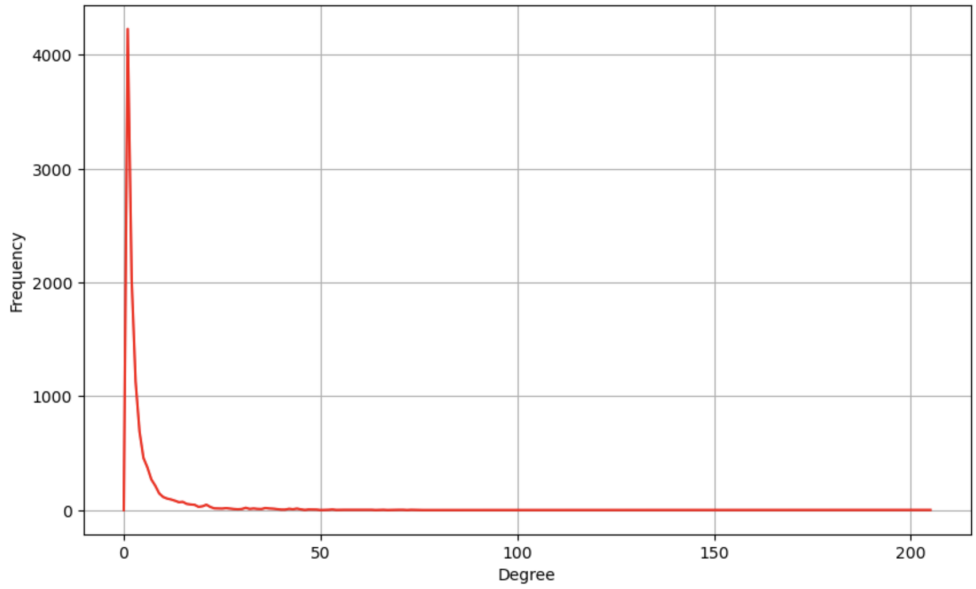


Figure 3.8: Degree distribution in Wikipedia-election dataset

3.6.2 Experimental Platform

Experiments are run on a Linux server of Xeon(R) Platinum 8380 CPU containing 160 cores and 2 threads per core, resulting in a total of 320 logical processing units and 1 Terabyte of DDR4 RAM.

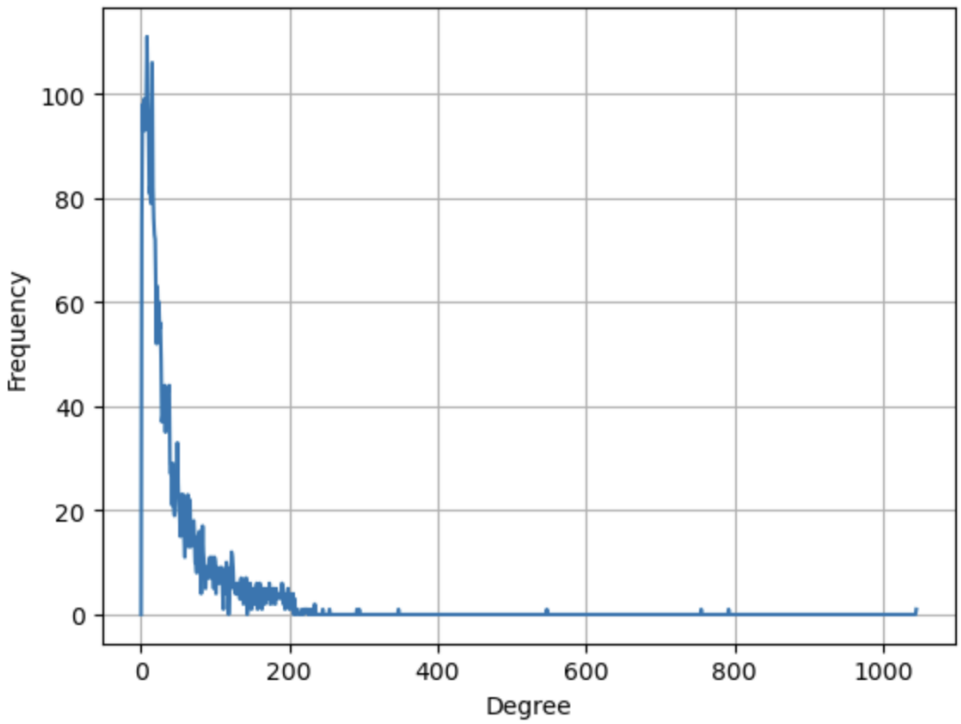


Figure 3.9: Degree distribution in Facebook dataset

3.6.3 Accuracy Experiments

This section presents the accuracy evaluations of our proposed estimates. We have generated a synthetic dataset as previously detailed and established a streaming simulation via Apache Kafka. The simulation architecture consists of a producer that streams edge data and a consumer that processes this data to estimate the number of triangles present. We employ a sampling technique for each analytical window to validate our estimates, constructing a graph that adheres to the predefined edge probabilities. On these graphs, we implement a brute-force approach to counting triangles, providing an empirical baseline for comparative analysis with our algorithmic estimates.

The results, tabulated in Table 3.1, present the accuracy of our estimates across various window sizes and slide intervals. In our case, the accuracy is calculated by:

$$\text{Accuracy } = 100 \cdot \left(1 - \frac{|\text{estimated \# triangles} - \text{total \# triangles}|}{\text{total \# triangles}}\right)$$

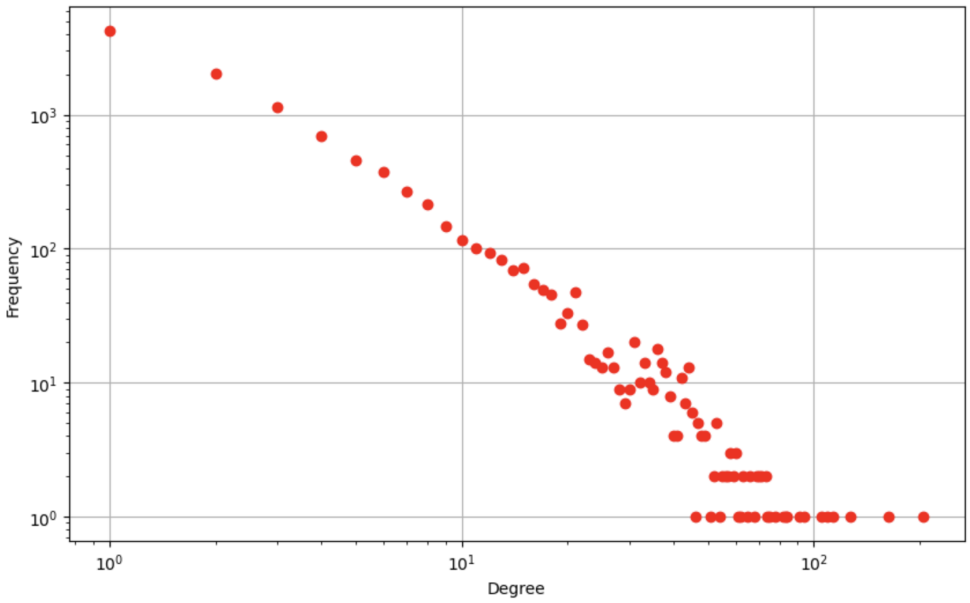


Figure 3.10: log-log graph of degree distribution in Wikipedia-election dataset

As can be observed, when the slide interval approaches the window size (a high slide-to-window size ratio), the accuracy of the estimates is reduced. This decline in precision could be attributed to the compression of the distribution that occurs when we divide the triangle counting function f by the number of vertices n to generate g to satisfy the Lipschitz condition. As n grows, our estimates potentially diverge more from the actual values. Conversely, excessively small ratios also yield subpar results, likely due to the breach of the Law of Large Numbers that our assumptions rest upon.

Optimal outcomes are observed when the slide interval is approximately 20% of the window size, suggesting a balanced slide-to-window ratio conducive to accurate estimations.

It is important to note that, as our model represents a novel approach that has not been previously studied, we lack an established baseline or gold standard for direct comparison. This inherent limitation means that our results must be interpreted within the context of new research rather than benchmarked against existing solutions.

The primary objective of this thesis is to demonstrate the feasibility of constructing a predictive model capable of yielding reasonably good results with minimal latency. The assertion of "satisfactory" performance pertains to our ability to achieve these results within the constraints and goals set forth by our research.

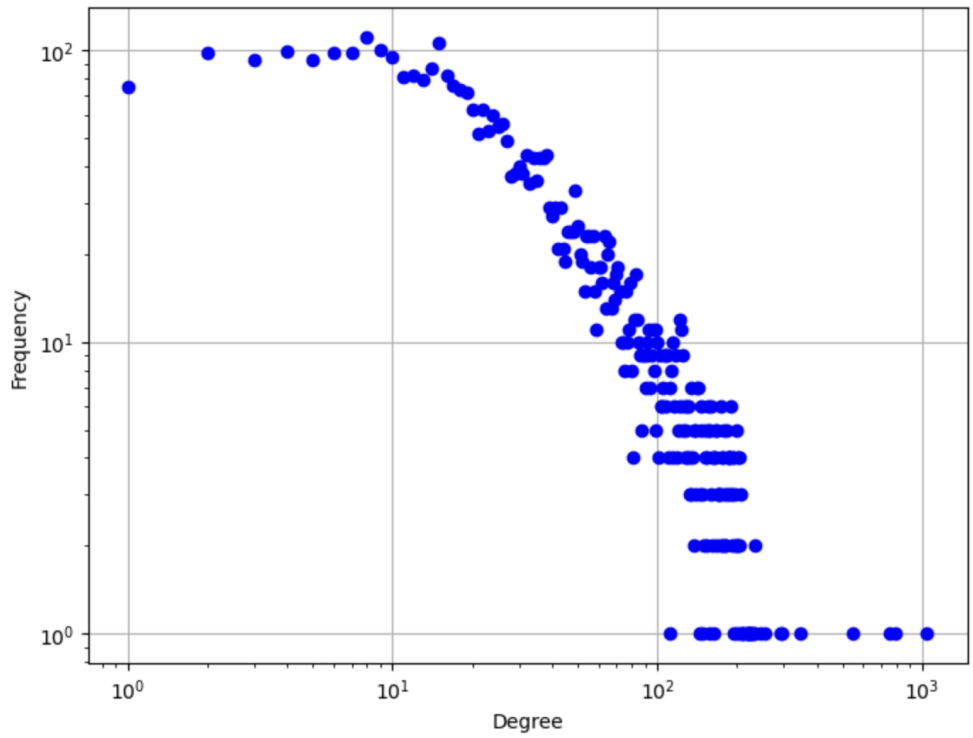


Figure 3.11: log-log graph of degree distribution in Facebook dataset

When evaluating accuracy, it is essential to consider the specific context and objectives of the study. In practical applications, acceptable or satisfactory levels of accuracy can vary significantly based on the use case, domain requirements, and the relative importance of latency versus precision. In our case, the emphasis was placed on achieving a balance between prediction quality and computational efficiency.

The absence of a baseline means that our findings should be regarded with caution. However, the fact that our model performs reasonably well and meets the intended low-latency criteria is a promising indicator of its potential utility. Future research could build upon our work to establish comparative baselines and further refine the model, thereby providing a more comprehensive assessment of its effectiveness.

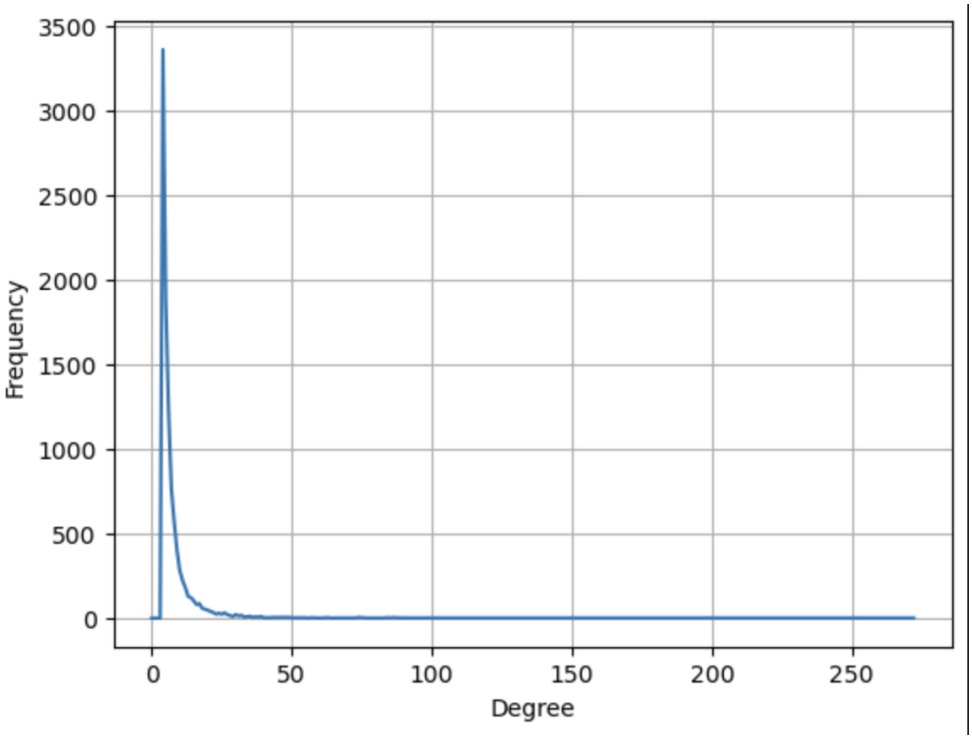


Figure 3.12: Degree distribution in our synthetic dataset

3.6.4 Latency Experiments

Conducting latency experiments is unnecessary. Since our methodology revolves purely around computational calculations without the need for intensive graph traversal, the calculation times are very small. We report wall clock time measurements for latency as in Table 3.1.

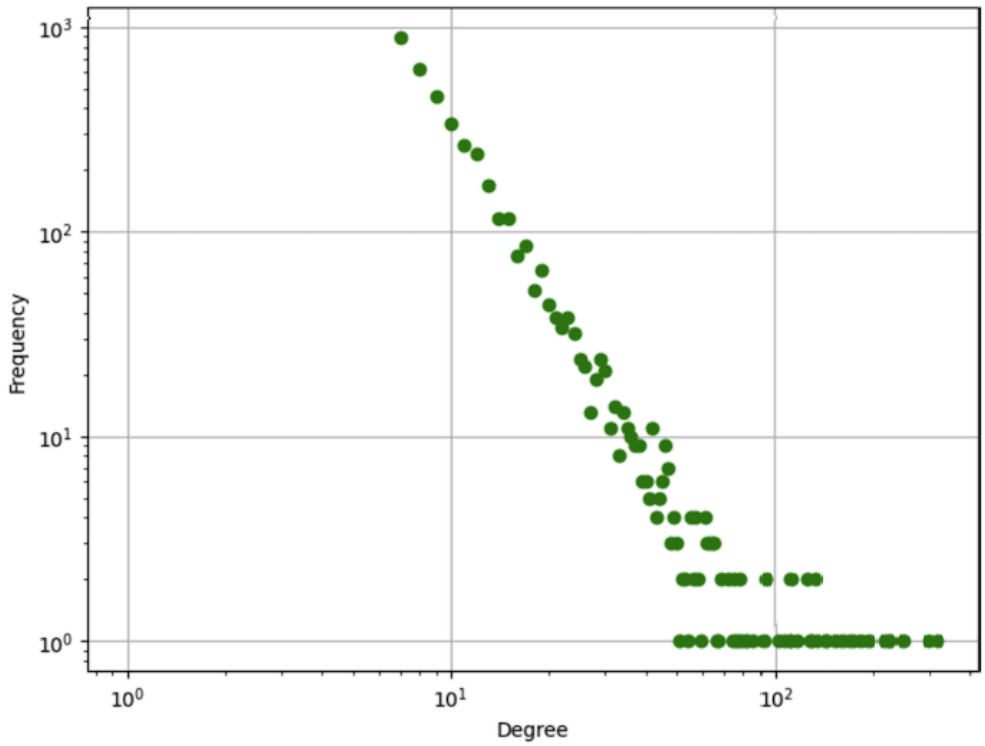


Figure 3.13: log-log graph of degree distribution in our synthetic dataset

	Slide interval		
Window Size	5 mins	10 mins	20 mins
30 mins	64.55	63.89	34.31
45 mins	65.05	66.92	57.79
60 mins	70.37	71.26	71.44
90 mins	47.22	59.91	71.26
120 mins	39.88	51.34	69.98

Table 3.1: Accuracy results

Chapter 4

Label Uncertain Streaming Graphs

The task of link prediction has received considerable attention across various fields, such as data mining, machine learning, recommendation systems, and network science [35, 23, 54]. The objective is to predict future connections within a graph, utilizing methodologies that range from neighbourhood-based approaches to matrix factorization and supervised learning despite the constraints posed by the streaming data. This chapter presents a novel link prediction algorithm for graph streams, emphasizing the design of probabilistic structures that enable real-time prediction with accuracy comparable to more traditional, computation-intensive methods.

The subsequent sections of this chapter discuss the structure and components of the graph edge-label prediction problem within the context of streaming graphs. First, we clearly define the streaming label prediction problem, followed by an in-depth exploration of core, neighbourhood-oriented link prediction metrics, specifically the Common Neighbour and Jaccard coefficient [35]. These metrics serve as pivotal target measures in our analysis of streaming label prediction.

To tackle the label prediction challenge comprehensively, we partition it into two distinct yet interconnected subproblems:

1. **Estimating the Future Graph Structure:** The initial step involves estimating the number of vertices that will appear in the forthcoming data window. Leveraging insights from Section 3.2, we establish an upper limit on the count of new vertices expected to materialize in the upcoming window, providing a predictive outlook on the graph's expansion (Section 4.1).

2. **Predictive Modeling Using Graph Neural Networks (GNNs):** With an estimated graph structure at hand, we employ a sophisticated GNN-based framework for link-label prediction, aiming to foresee potential connections within the forthcoming window. This model integrates learned graph features and topological information to effectively predict future edge formations and their labels. Figure 4.1 visually summarizes our approach (Section 4.2).

Together, these subproblems form a comprehensive methodology for addressing the dynamic and multifaceted nature of the streaming label prediction problem, integrating data from past observations with advanced predictive algorithms to inform and refine future predictions. We discuss them individually in the subsequent sections.

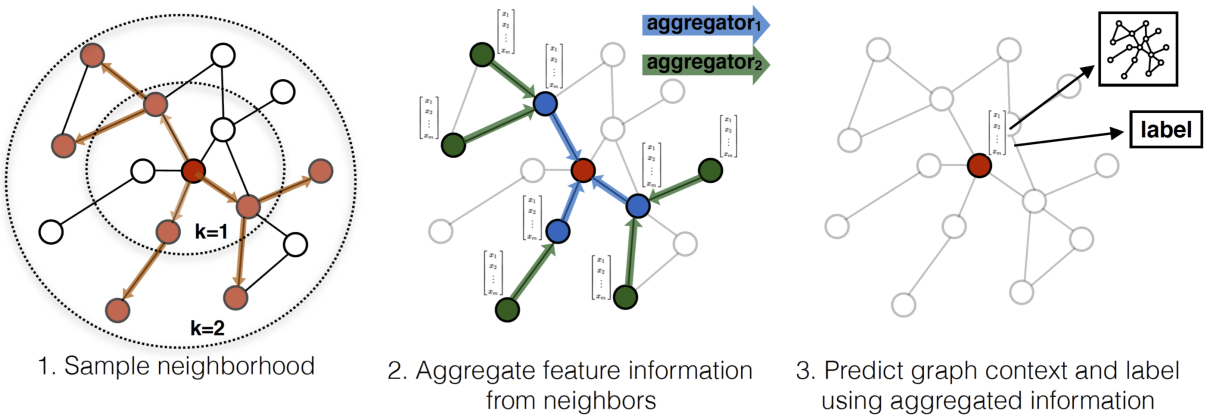


Figure 4.1: The GraphSAGE aggregate approach [23]

4.1 Definitions and Target Measures

We first start with a slightly revised definition of graph snapshot that takes into account the uncertainty.

Definition 17 (Graph Snapshot)**.** *At any given time t, a graph snapshot G_k is a four-tuple $(V_k, E_k, L_k, \Sigma_k)$ where k, V_k, E_k are the same as defined in Definition 13 and Σ_k is the set of distinct edge-labels seen the window k and $L_k : E_k \to \Sigma_k$ is a function that associates labels to the edges.*

Next, we explore two basic neighbourhood-based graph proximity measures, which serve as critical metrics for assessing the efficacy of our label prediction model. These measures are fundamental to the process of label prediction, encapsulating the core objectives our model seeks to achieve. Throughout this chapter, our model strives to optimize these metrics, treating them as benchmark standards for the task at hand. We henceforth refer to these graph measures as **target measures** for our analysis of streaming label prediction.

Define a set $\tau(u, k)$ such that it contains all the vertex-label pairs incident to $u \in V_k$ for the graph snapshot $\mathcal{G}_k$. Taking ideas from [35], we define the following target measures:

- **Common Neighbour:** The concept of "Common Neighbour" suggests that two vertices x and y in a graph are more likely to form a link in the future if they share a significant number of neighbours, represented by the overlap of their neighbour sets $\tau(x, k)$ and $\tau(y, k)$. This idea is grounded in the observation that shared neighbours often imply a higher chance of interaction or connection. For instance, in social networks, two people with many mutual friends are more likely to meet and establish a connection. The most straightforward implementation of this concept in link prediction algorithms uses the score function defined by the intersection of the neighbour sets of the two vertices, $\text{score}(x, y) := |\tau(x) \cap \tau(y)|$, which quantifies the number of common neighbours. We use this measure as a heuristic while implementing our machine-learning model.

  Formally,

  Definition 18 (Common Neighbour)**.** *For any* $u, v \in V_k$

  $$CN(u, v) = |\tau(u, k) \cap \tau(v, k)|$$

- **Jaccard Coefficient:** The "Jaccard Coefficient" is another popular metric used predominantly in information retrieval to measure the similarity between two sets. It evaluates the likelihood that both vertices x and y share a feature f that one of them has. In the context of graph-based models, where features are akin to the labels incident to vertices in a graph snapshot, the Jaccard Coefficient is calculated as $\text{score}(x, y) := |\tau(x) \cap \tau(y)|/|\tau(x) \cup \tau(y)|$. This ratio provides a normalized measure of similarity, ranging between zero (no commonality) and one (identical sets), thus offering a more nuanced assessment compared to the absolute count of common neighbours.

  Formally,

Definition 19 (Jaccard Coefficient)**.** *For any* $u, v \in V_k$

$$JC(u, v) = \frac{|\tau(u, k) \cap \tau(v, k)|}{|\tau(u, k) \cup \tau(v, k)|}$$

4.2 Label Prediction

In the context of streaming graphs, predicting the new incoming vertices, edges, and their corresponding attributes for subsequent windows requires a preliminary assessment of the active vertex count expected in the upcoming window. As discussed in Section 3.2, we have already formulated an approximation for the count of active vertices. This estimation will be integrally applied within the current analytical framework. With an estimation of the vertex population at hand, the next logical step is to predict the incoming edges and their corresponding labels.

To achieve this, we devise a Graph Neural Network (GNN) model, drawing conceptual inspiration from the GraphSage module found within the NetworkX library. The algorithm at the core of this model is designed to compute vertex embeddings by sequentially navigating through the neighbourhood of each anticipated vertex within a given graph snapshot. This process effectively performs an 'aggregation' of the embeddings from adjacent vertices and uses them to calculate the embedding of the current vertex. Subsequently, these embeddings are utilized to formulate predictions for the ensuing window. When the expected window actually arrives, we use this data to retrain the model and update the embeddings.

This iterative methodology is crucial for capturing the local structural intricacies and edge attributes, i.e., each vertex embedding is calculated such that it encompasses the characteristics of its neighbourhood. The model adeptly uses the collective characteristics and interrelations of neighbourhood vertices through its aggregation mechanism, thereby fortifying the predictive modelling of vertex embeddings. The process of associating these labels has a computational complexity of $O(n^2)$, where $n = |V_k|$ and V_k represents the set of vertices within a particular graph snapshot $\mathcal{G}_k = (V_k, E_k, L_k)$.

In the remainder, we discuss the vertex embedding generation mechanism using the aggregation methodology and how we use these embeddings to predict the edge label pairs in the next window. Here, we assume that the estimate provided in Section 3.2 gives us the expected vertices in the new window, and we use it to generate embeddings and make predictions.

4.2.1 Embedding generation

In this section, we will explore the technique for creating vertex embeddings with an aggregation method and then detail how we use these vertex representations to forecast connections and their labels in future network snapshots. This discussion is based on the vertex count projection found in Section 3.2, which we'll leverage to craft embeddings and perform predictions.

Building upon our earlier discussion, we create a vertex's embedding by combining information from neighbouring vertices and the associated edge labels. For a graph snapshot denoted as $\mathcal{G}_k = (V_k, E_k, L_k, \Sigma_k)$, we assign a sequential rank to all labels encountered to date, labelling each $l \in \Sigma_k$ with a unique number from 1 to $|\Sigma_k|$. This numbering stays consistent in all windows. Denote the embedding of vertex u at iteration i as e_u^i. At iteration i for vertex u in $\mathcal{G}_k$, we follow the same updating procedure proposed by Hamilton et al. [23]:

$$m_u^i = W_i \cdot h_u^{i-1} + W_r \cdot AGG(\{m_v^{i-1} + L(u,v),\ \forall v \in N(u)\})$$

In this equation, W_i and W_r are matrices that the learning process will fine-tune, and $N(u)$ lists all neighbouring vertices to u. The function $L(u,v)$ specifies the numerical identifier for the label of the edge between vertices u and v.

The AGG function aggregates input from the vertex's neighbours and their edge labels. To keep our model straightforward, we use a mean aggregation approach, defined as:

$$AGG(\{m_v^{i-1} + L(u,v),\ \forall v \in N(u)\}) = \frac{1}{|N(u)|} \sum_{v \in N(u)} m_v^{i-1} + L(u,v)$$

Integrating all the elements, we describe the embedding generation Algorithm 2.

Note that the aggregation step at the i^{th} iteration depends on the representations generated in the previous (i.e., $i - 1^{th}$) iteration. Putting it all together, the embedding generation model can be visualized as shown in Figure 4.2. We use the GraphSage module in the NetworkX library to implement this in our model.

4.2.2 Label Classification

Upon the stabilization of the algorithm, which is indicated by the convergence of the embeddings, we proceed to utilize them to estimate the likelihood of a connection between pairs of vertices, along with identifying the edge label of such potential connection. This

Algorithm 2: Embedding generation algorithm

Input : Graph Snapshot $\mathcal{G}_k = (V_k, E_k, L_k, \Sigma_k)$; input features $\{x_u, \forall u \in V_k\}$; depth Q; weight matrices $W^q, \forall q \in \{1, \ldots, Q\}$; neighbourhood function $\mathcal{Q} : u \rightarrow 2^{V_k}$

Output: Vector representations z_u for all $u \in V_k$

$m_u^0 \leftarrow x_u, \forall u \in V_k$;
for $q = 1 \ldots Q$ **do**
  for $u \in V$ **do**
    $m_u^i \leftarrow \mathrm{AGG}_i(\{m_v^{i-1} + L(u, v), \forall v \in \mathcal{N}(u)\})$
  end
  $m_u^i \leftarrow m_u^i / \|m_u^i\|_2, \forall u \in V_k$
end
$z_v \leftarrow m_u^i, \forall u \in V_k$

predictive phase involves evaluating pairs of vertex embeddings and deducing the numerical attribute that signifies the nature of the connection. Essentially, for every pair of vertices, if an edge between them exists, the model returns the numeric value of the label that is most likely to be associated with it and 0 if the edge is not expected to exist. We use a Multi-Layer Perceptron (MLP) [**?**], which takes in the element-wise product of the embedding vector for a pair of vertices and outputs the type of label expected between them.

$$\mathrm{Label}(u, v) = MLP(m_u \odot m_v)$$

This label association step takes $O(n^2)$ where $n = |V_k|$ and V_k is the vertex set in a particular graph snapshot $\mathcal{G}_k = (V_k, E_k, L_k)$, as we need to iterate over every pair of vertices in the graph snapshot.

Further, in order to learn the weight matrices in an unsupervised setting, we apply a graph-based loss function that minimizes the error in prediction.

$$\text{Loss } = -\text{mean}(\log(\text{correct predictions} + \epsilon)) + \text{mean}(\log(1 - \text{incorrect predictions} + \epsilon))$$

where $\epsilon = 1e - 10$ is used for numeric stability.

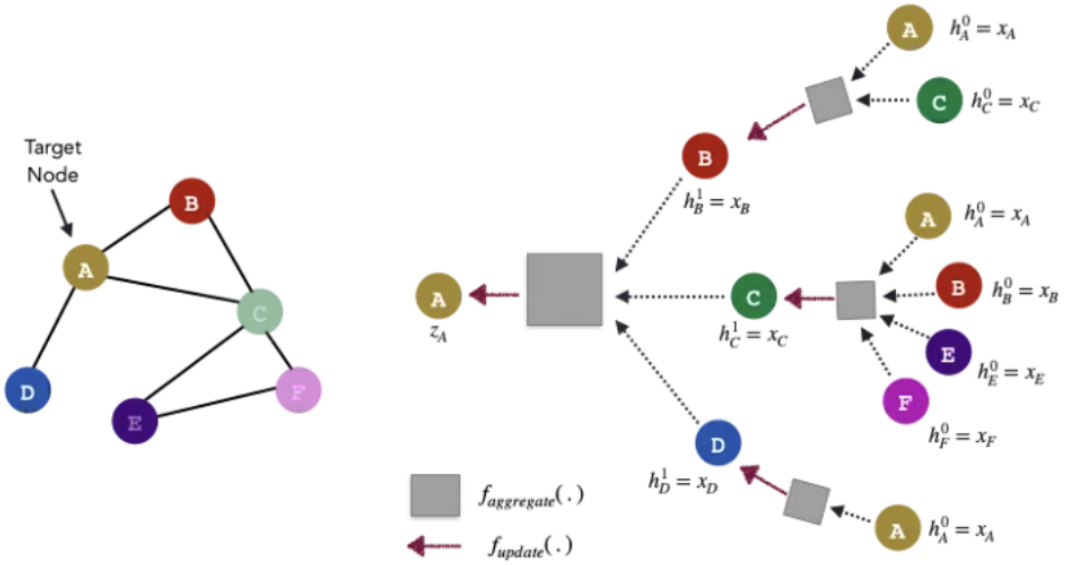


Figure 4.2: The message passing mechanism. Image source: CS224 slides Stanford University

4.3 Experimental evaluation

In this section, we present the experimental evaluation of the discussed predictive GNN model. Following the methodology discussed in Section 3.6, we generate a synthetic graph dataset, which follows a preferential attachment policy [44] and associate labels to them randomly. Similar to Section 3.6, we employ the Barabási–Albert (BA) model, a renowned method for generating random networks that inherently exhibit scale-free properties, and associate labels to the generated graph randomly.

To emulate the streaming environment, we use Apache Kafka to set a producer and consumer instance where the producer dispatches streaming edges, and then the consumer ingests this data and runs the GNN module to predict labels in the next window. Similar to the previous experiments, we use a sliding window-based management system to do our computations.

4.3.1 Experimental Platform

Experiments are run on a Linux server of Xeon(R) Platinum 8380 CPU containing 160 cores and 2 threads per core, resulting in a total of 320 logical processing units and 1 Terabyte of DDR4 RAM.

4.3.2 Accuracy Experiments

We consider a synthetic dataset with a total 5 labels for the first set of experiments. The corresponding vertex degree distribution and edge label distribution for our dataset are presented in Figure 4.3 and Figure 4.4.

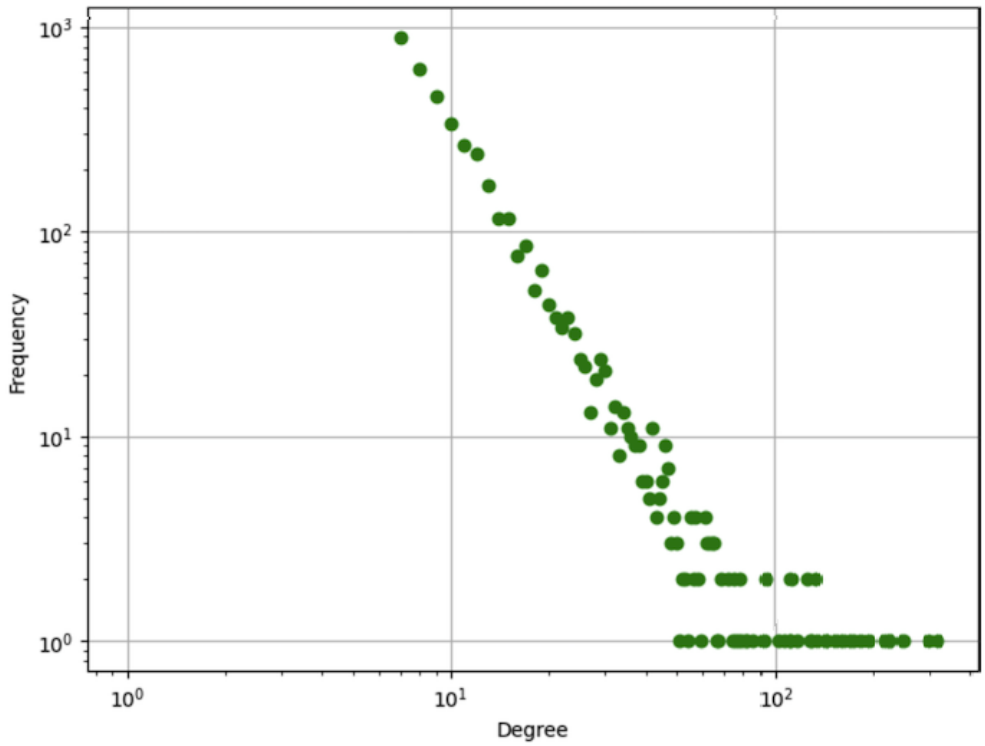


Figure 4.3: log-log graph of degree distribution in the synthetic dataset with 5 labels

Given this, we depict in Figure 4.5 the accuracy of the prediction of our model along with the baseline. It is interesting to see that after seeing about 57 windows, we start getting stable results.

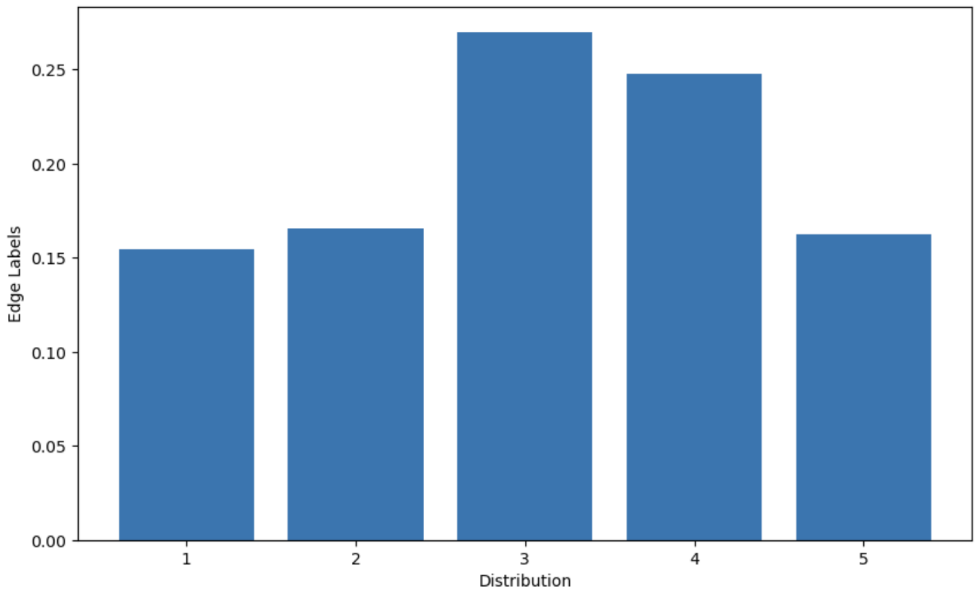


Figure 4.4: Edge label distribution in the synthetic dataset with 5 labels

We repeat the experiments for synthetic datasets with a higher number of distinct labels. We generate a dataset with $100,000$ vertices, with 10 and 15 edge labels. Figure 4.6 and Figure 4.7 show the vertex degree distribution and edge label distribution for the 10-label case and Figure 4.8 and Figure 4.9 show the vertex degree distribution and edge label distribution for the 15-label case.

Figure 4.10 and Figure 4.11 represent the label prediction accuracy for the 10 and 15 label cases respectively. Notice that in both cases, the accuracy is pretty much stable after around 100 windows, and 80% of the edge labels are being predicted correctly.

4.3.3 Latency Experiments

Similar to Section 3.6, conducting latency experiments are unnecessary. Since our methodology is purely predictive, i.e., we are interested in the next window, the objective is to finish the computation before the window materializes.

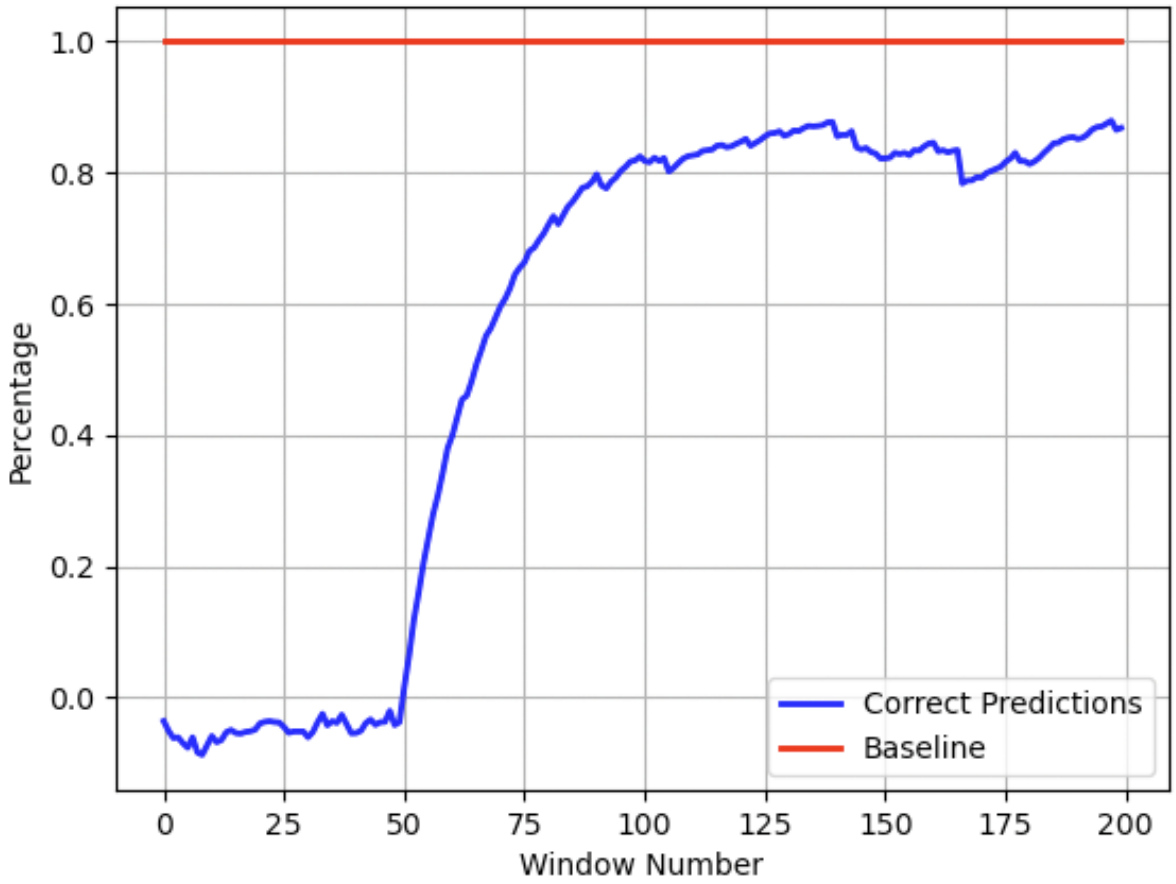


Figure 4.5: Accuracy of prediction algorithm with 5 labels

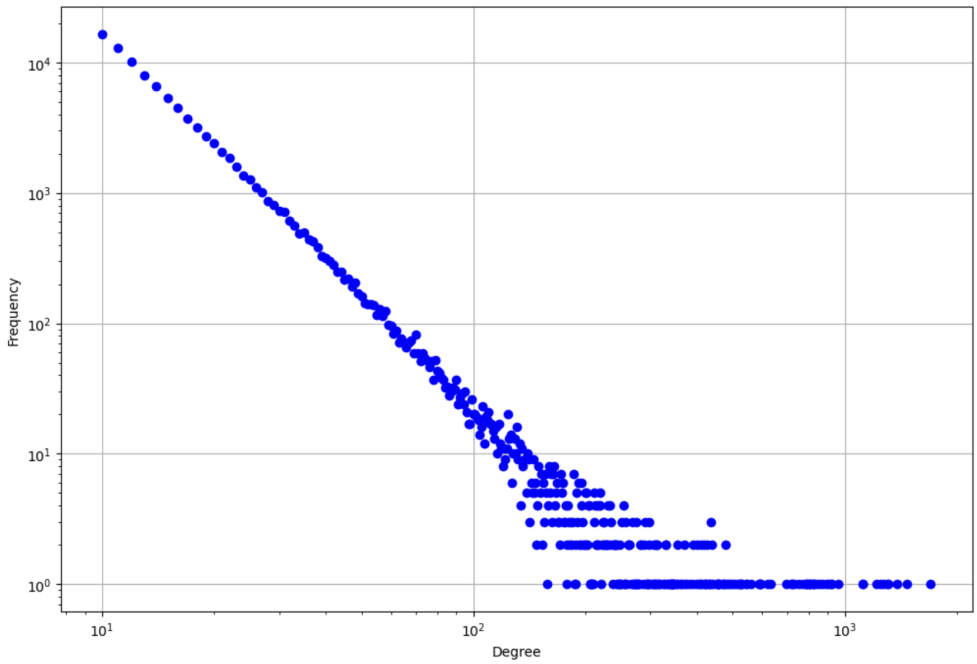


Figure 4.6: log-log graph of degree distribution in the synthetic dataset with 10 labels

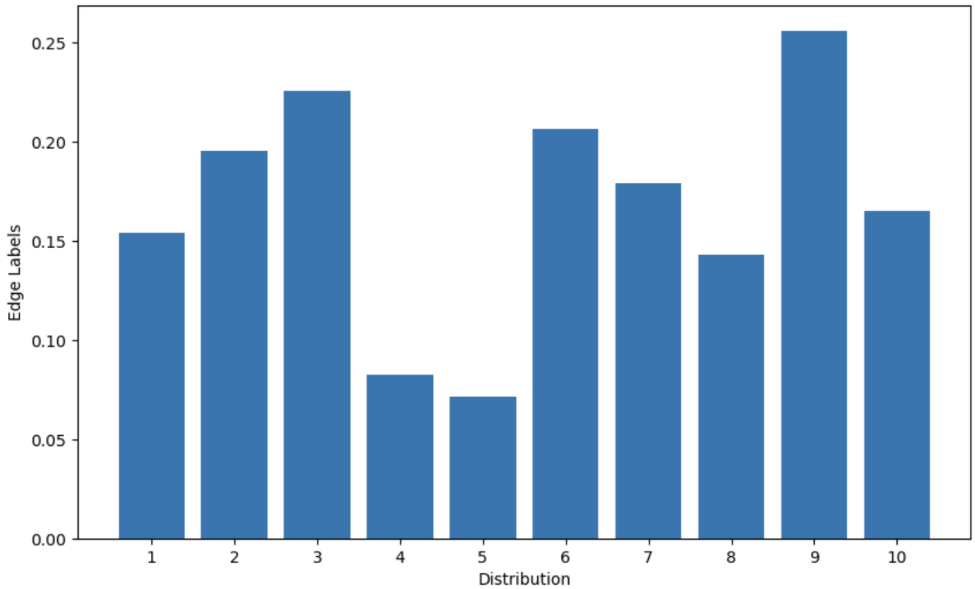


Figure 4.7: Edge label distribution in the synthetic dataset with 10 labels

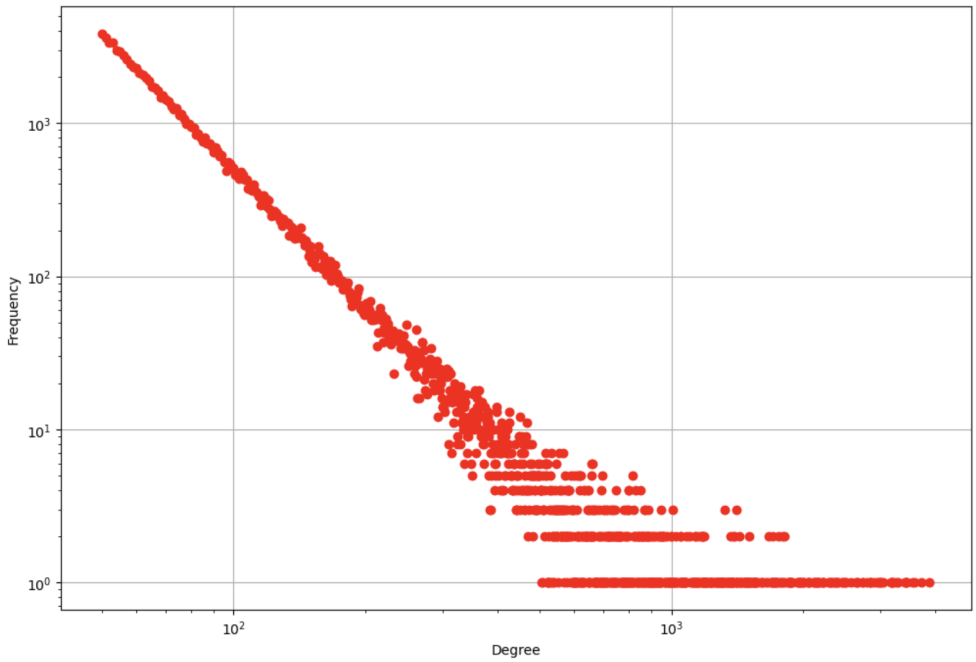


Figure 4.8: log-log graph of degree distribution in the synthetic dataset with 10 labels

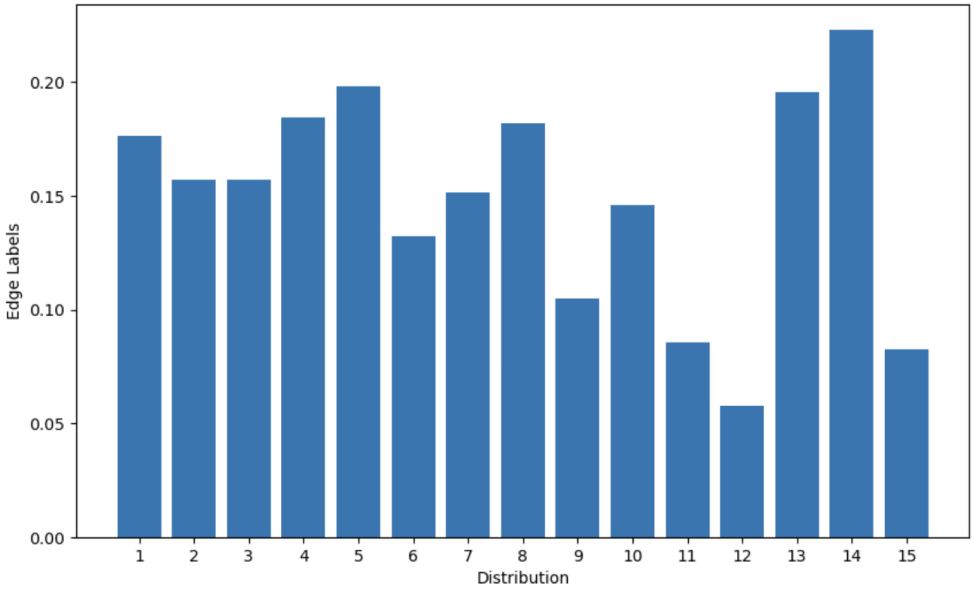


Figure 4.9: Edge label distribution in the synthetic dataset with 10 labels

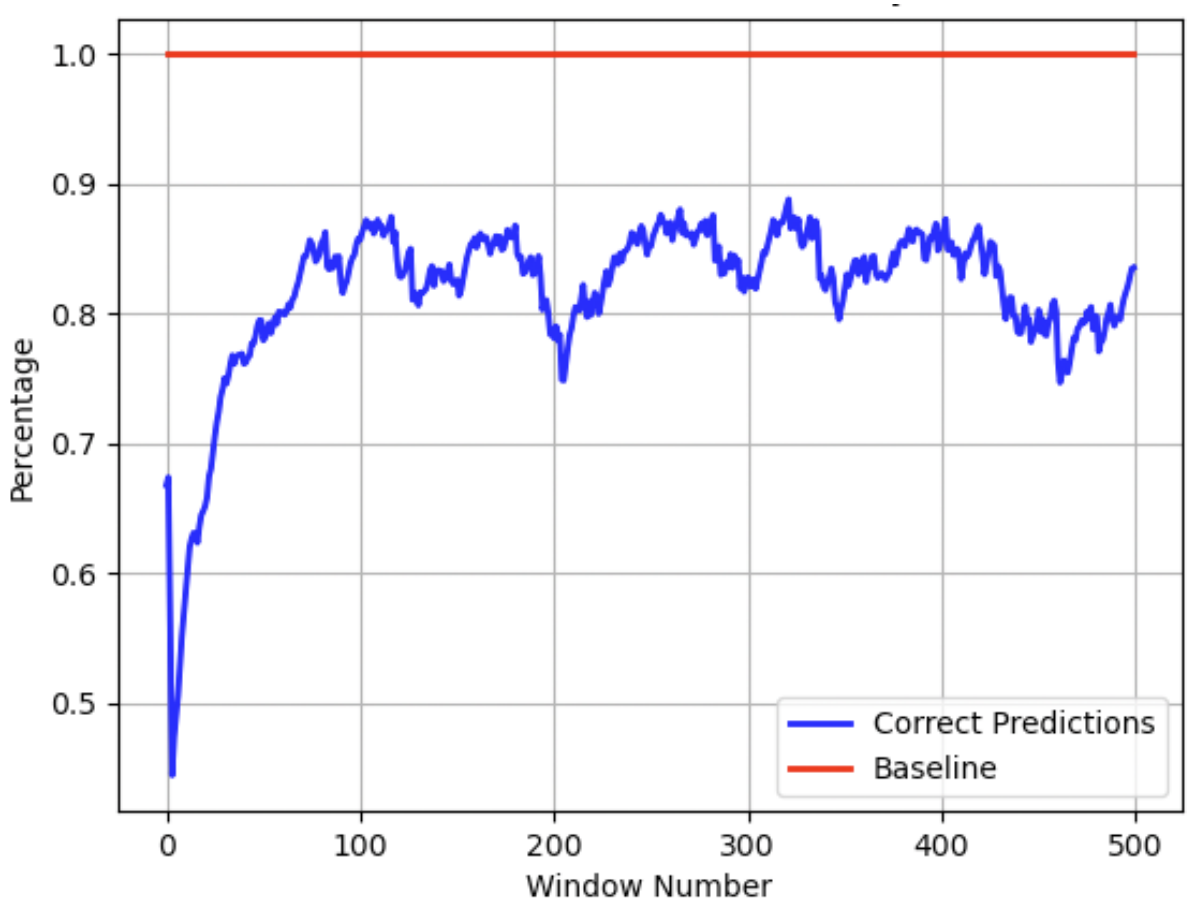


Figure 4.10: Accuracy of prediction algorithm with 10 labels

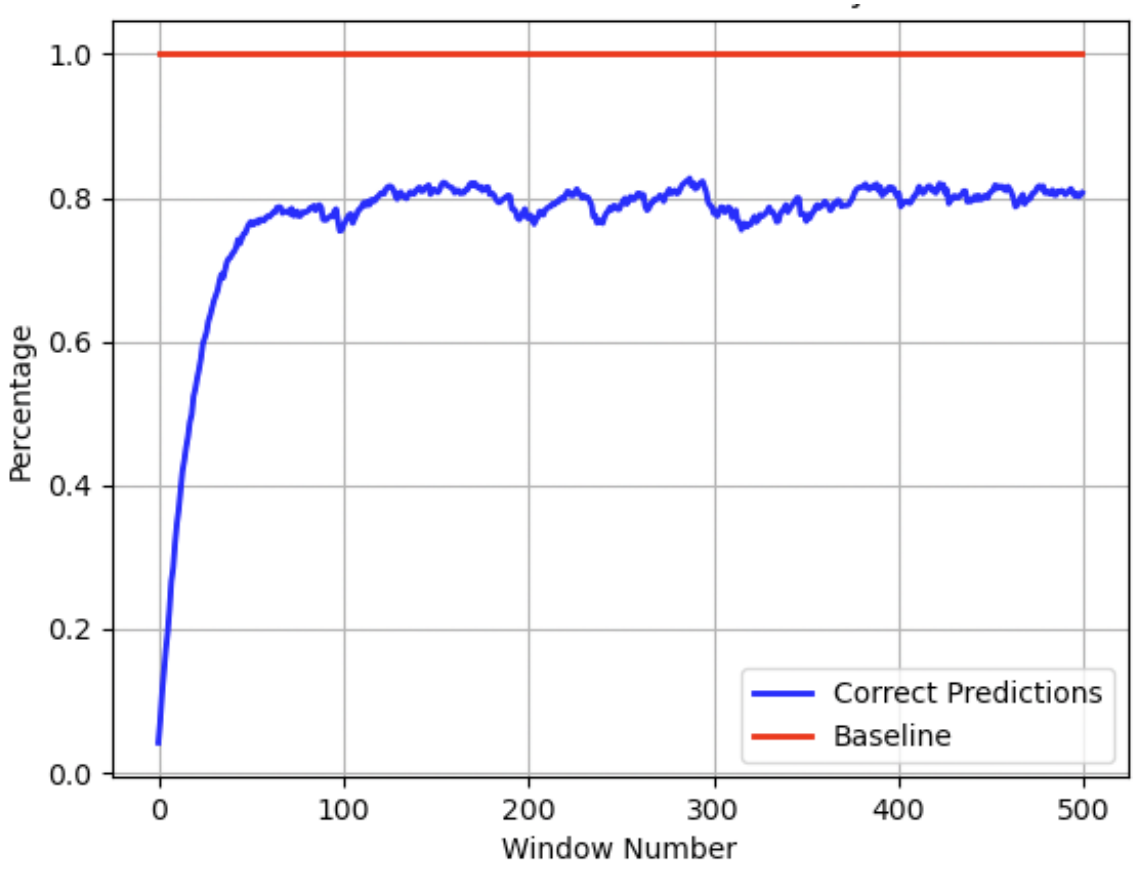


Figure 4.11: Accuracy of prediction algorithm with 15 labels

Chapter 5

Conclusion and Future Work

Graph database systems are being extensively used to manage complex, interconnected data across various sectors, including social networks, transportation networks, and genomic databases, among others. Often, this data is dynamic, characterized by incomplete or probabilistic connections that pose significant challenges to data processing and analysis. This thesis introduces a novel framework designed to incorporate the management of uncertainty within streaming graphs, significantly boosting capabilities for real-time decision-making and analytical processes.

The focus of this research primarily lies in addressing uncertainties related to edges and labels within these graphs. We delve into specific challenges in each area, developing comprehensive strategies to tackle these complexities effectively. First, we explore the problem of approximate triangle counting in edge uncertain streaming graphs. For this, we adopt a martingale-based approach, which allows for precise estimation of triangle numbers within specific graph windows.

Subsequently, we address label prediction in attribute uncertain streaming graphs. Here, we implement a framework based on Graph Neural Networks (GNNs), which is finely tuned to adapt to the evolving structure of the graph and accurately predict forthcoming labels.

Through rigorous experimental validation, we demonstrate the effectiveness of both the martingale-based approach for triangle counting and the GNN-based method for label prediction. The results from these experiments are promising, showing that the methodologies developed not only address the challenges posed by uncertain and dynamic data but also perform exceptionally well in real-world scenarios.

5.1 Future Work

While the probabilistic modeling method deployed in this thesis has achieved satisfactory results across the datasets tested, yet significant potential exists for further refinement and expansion. A notable avenue for future research is the extension of the martingale-based method to encompass larger subgraphs, including those with a greater number of vertices. Although the Lipschitz condition's applicability becomes limited as the subgraph size increases, it would be insightful to determine the threshold beyond which the approximation yields diminishing returns.

Additionally, our model could be expanded beyond subgraph counting to address reachability queries. Specifically, it would be interesting to explore if and how vertices may connect over time within the probabilistic framework of our model. Another intriguing aspect to consider is the influence maximization problem within streaming windows, which could potentially enhance the understanding of influence dynamics over temporal graphs compared to static analyses.

The majority of existing work on uncertain graphs relies heavily on sampling multiple independent possible worlds to estimate the likelihood of specific properties. An innovative research direction might involve identifying a single "optimal" possible world that sufficiently preserves the inherent properties of the graph. This approach would challenge the traditional multiple-sampling strategies and could redefine how uncertain graphs are processed.

Given the challenges of exact computation in large-scale uncertain graphs, there is a fundamental trade-off between scalability and efficiency versus accuracy. Identifying specific application areas and their particular needs is crucial—such as the balance between efficiency and accuracy or the trade-offs between false positive and false negative rates. Additionally, understanding the costs associated with probing uncertain edges and fine-tuning algorithm-specific parameters to optimize outcomes remains a critical area for development. For example, applications that require highly precise identification of reliable entities might benefit from a reliable path-based method rather than more computationally intensive sampling techniques. Each of these directions offers the potential to advance the field and tailor uncertain graph processing techniques to meet the diverse demands of real-world applications better.

References

[1] Ehab Abdelhamid, Mustafa Canim, Mohammad Sadoghi, Bishwaranjan Bhattacharjee, Yuan-Chi Chang, and Panos Kalnis. Incremental frequent subgraph mining on large evolving graphs. In *Proc. 2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1767–1768, 2018.

[2] Eytan Adar and Christopher Ré. Managing uncertainty in social networks. *IEEE Data Eng. Bull.*, 30:15–22, 01 2007.

[3] Charu Aggarwal and Philip Yu. A survey of uncertain data algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering*, 21:609 – 623, 06 2009.

[4] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley, New York, second edition, 2004.

[5] Cigdem Aslay, Muhammad Anis Uddin Nasir, Gianmarco De Francisci Morales, and Aristides Gionis. Mining frequent patterns in evolving graphs. In *Proc. 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 923–932, New York, NY, USA, 2018. Association for Computing Machinery.

[6] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. Topic-aware social influence propagation models. In *Proc. 2012 IEEE 12th International Conference on Data Mining*, pages 81–90, 2012.

[7] Shishir Bharathi, David Kempe, and Mahyar Salek. Competitive influence maximization in social networks. In *Internet and Network Economics*, pages 306–311, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[8] Paolo Boldi, Francesco Bonchi, Aristides Gionis, and Tamir Tassa. Injecting uncertainty in graphs for identity obfuscation. *Proc. VLDB Endow.*, 5(11):1376–1387, 2012.

[9] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, page 946–957, USA, 2014.

[10] Lei Chen and Changliang Wang. Continuous subgraph pattern search over certain and uncertain graph streams. *IEEE Transactions on Knowledge and Data Engineering*, 22(8):1093–1109, 2010.

[11] Xiaowei Chen and John C.S. Lui. A unified framework to estimate global and local graphlet counts for streaming graphs. In *Proc. 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, page 131–138, New York, NY, USA, 2017.

[12] James Cheng, Yiping Ke, Wilfred Ng, and An Lu. Fg-index: towards verification-free query processing on graph databases. In *Proc. 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, page 857–872, New York, NY, USA, 2007.

[13] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and distance queries via 2-hop labels. In *Proc. Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, page 937–946, USA, 2002.

[14] John B. Collins and Steven T. Smith. Network discovery for uncertain graphs. In *Proc. 17th International Conference on Information Fusion*, pages 1–8, 2014.

[15] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, page 57–66, New York, NY, USA, 2001.

[16] Krogan et al. Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, 440(7084):637–643, 2006.

[17] Wenfei Fan, Jianzhong Li, Jizhou Luo, Zijing Tan, Xin Wang, and Yinghui Wu. Incremental graph pattern matching. page 925–936, 2011.

[18] Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Yinghui Wu. Adding regular expressions to graph reachability and pattern queries. In *Proc. IEEE 27th International Conference on Data Engineering, ICDE'11*, pages 39–50, 2011.

[19] Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, Yinghui Wu, and Yunpeng Wu. Graph pattern matching: from intractable to polynomial time. *Proc. VLDB Endow.*, 3(1–2):264–275, 2010.

[20] George S. Fishman. A comparison of four monte carlo methods for estimating the probability of s-t connectedness. *IEEE Transactions on Reliability*, 35(2):145–155, 1986.

[21] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

[22] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[23] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proc. of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017.

[24] S.S. Haykin. *Neural Networks: A Comprehensive Foundation*. International edition. Prentice Hall, 1999.

[25] M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. IEEE 36th Annual Foundations of Computer Science*, pages 453–462, 1995.

[26] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Videograph: Recognizing minutes-long human activities in videos. In *Proc. ICCV Workshop on Scene Graph Representation and Learning*, 2019.

[27] Ruoming Jin, Lin Liu, Bolin Ding, and Haixun Wang. Distance-constraint reachability computation in uncertain graphs. *Proc. VLDB Endow.*, 4(9):551–562, 2011.

[28] Xiangyu Ke, Arijit Khan, and Leroy Lim Hong Quan. An in-depth comparison of s-t reliability algorithms over uncertain graphs. *Proc. VLDB Endow.*, 12(8):864–876, 2019.

[29] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proc. 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'03, page 137–146, 2003.

[30] Arijit Khan, Francesco Bonchi, Aristides Gionis, and Francesco Gullo. Fast reliability search in uncertain graphs. In *Proc. International Conference on Extending Database Technology, EDBT, Athens, Greece, March 24-28, 2014*, pages 535–546, 2014.

[31] Arijit Khan, Yuan Ye, and Lei Chen. *Introduction to Uncertain Graphs*, pages 1–10. Springer International Publishing, Cham, 2018.

[32] Theodoros Lappas, Evimaria Terzi, Dimitrios Gunopulos, and Heikki Mannila. Finding effectors in social networks. In *Proc. 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, page 1059–1068, 2010.

[33] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proc. 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'06, page 631–636, 2006.

[34] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[35] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proc. 12th International Conference on Information and Knowledge Management*, CIKM '03, page 556–559, 2003.

[36] Lin Liu, Ruoming Jin, Charu Aggarwal, and Yelong Shen. Reliable clustering on uncertain graphs. In *Proc. 12th IEEE International Conference on Data Mining*, pages 459–468, 2012.

[37] Paul Liu, Austin R. Benson, and Moses Charikar. Sampling methods for counting temporal motifs. In *Proc. 12th ACM International Conference on Web Search and Data Mining*, WSDM '19, page 294–302, 2019.

[38] Ming-ming Ma, Hao Chen, Xiu-ming Wang, and Xiao He. Inversion of shear velocity profile in a cased borehole. In *Proc. 2012 Symposium on Piezoelectricity, Acoustic Waves, and Device Applications*, pages 310–313, 2012.

[39] Walaa Eldin Moustafa, Angelika Kimmig, Amol Deshpande, and Lise Getoor. Subgraph pattern matching over uncertain graphs with identity linkage uncertainty. In *Proc. 30th IEEE International Conference on Data Engineering*, pages 904–915, 2014.

[40] Pacaci, Anil. *Models and Algorithms for Persistent Queries over Streaming Graphs*. PhD thesis, University of Waterloo, 2022.

[41] Panos Parchas, Nikolaos Papailiou, Dimitris Papadias, and Francesco Bonchi. Uncertain graph sparsification. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2435–2449, 2018.

[42] Michalis Potamias, Francesco Bonchi, Aristides Gionis, and George Kollios. k-nearest neighbors in uncertain graphs. *Proc. VLDB Endow.*, 3(1–2):997–1008, 2010.

[43] Stephan Seufert, Avishek Anand, Srikanta Bedathur, and Gerhard Weikum. Ferrari: Flexible and efficient reachability range assignment for graph indexing. In *Proc. 29th IEEE International Conference on Data Engineering*, pages 1009–1020, 2013.

[44] Sheshbolouki, Aida. *Mining Butterflies in Streaming Graphs*. PhD thesis, University of Waterloo, 2023.

[45] Suprosanna Shit, Rajat Koner, Bastian Wittmann, Johannes Paetzold, Ivan Ezhov, Hongwei Li, Jiazhen Pan, Sahand Sharifzadeh, Georgios Kaissis, Volker Tresp, and Bjoern Menze. Relationformer: A unified framework for image-to-graph generation. In *Proc. 17th European Conference on Computer Vision*, page 422–439, 2022.

[46] Jiao Su, Qing Zhu, Hao Wei, and Jeffrey Xu Yu. Reachability querying: Can it be even faster? *IEEE Transactions on Knowledge and Data Engineering*, 29(3):683–697, 2017.

[47] Juliane Verwiebe, Philipp M. Grulich, Jonas Traub, and Volker Markl. Survey of window types for aggregation in stream processing systems. *The VLDB Journal*, 32(5):985–1011, 2023.

[48] Sarisht Wadhwa, Anagh Prasad, Sayan Ranu, Amitabha Bagchi, and Srikanta Bedathur. Efficiently answering regular simple path queries on large labeled networks. In *Proc. 2019 International Conference on Management of Data*, SIGMOD '19, page 1463–1480, 2019.

[49] Pinghui Wang, John C.S. Lui, Don Towsley, and Junzhou Zhao. Minfer: A method of inferring motif statistics from sampled edges. In *Proc. 32nd IEEE International Conference on Data Engineering*, pages 1050–1061, 2016.

[50] Wikipedia. Martingale (betting system) — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Martingale%20(betting%20system)&oldid=1217696798, 2024. [Online; accessed 09-April-2024].

[51] Xifeng Yan, Philip S. Yu, and Jiawei Han. Graph indexing: a frequent structure-based approach. In *Proc. ACM SIGMOD International Conference on Management of Data*, SIGMOD'04, page 335–346, 2004.

[52] Ye Yuan, Guoren Wang, Lei Chen, and Haixun Wang. Efficient subgraph similarity search on large probabilistic graph databases. *Proc. VLDB Endow.*, 5(9):800–811, 2012.

[53] Ye Yuan, Guoren Wang, Haixun Wang, and Lei Chen. Efficient subgraph search over large uncertain graphs. *Proc. VLDB Endow.*, 4(11):876–886, 2011.

[54] Hao Zhou, Anna A. Shaverdian, H. V. Jagadish, and George Michailidis. Querying graphs with uncertain predicates. In *Proc. 8th Workshop on Mining and Learning with Graphs*, MLG '10, page 163–170, 2010.

[55] Rong Zhu, Zhaonian Zou, and Jianzhong Li. Top-k reliability search on uncertain graphs. In *Proc. IEEE International Conference on Data Mining*, pages 659–668, 2015.

[56] Lei Zou, Lei Chen, and M. Tamer Özsu. Distance-join: pattern match query in a large graph database. *Proc. VLDB Endow.*, 2(1):886–897, 2009.

[57] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. Mining frequent subgraph patterns from uncertain graph data. *IEEE Transactions on Knowledge and Data Engineering*, 22(9):1203–1218, 2010.