Estimate and Reduce Uncertainty in Uncertain Graphs

Naheed Anjum Arafat
Nanyang Technological University
Singapore
naheed_anjum@u.nus.edu

Ehsan Bonabi Mobaraki
Aalborg University
Denmark
ebmo@cs.aau.dk

Arijit Khan
Aalborg University
Denmark
arijitk@cs.aau.dk

Yllka Velaj
University of Vienna
Austria
yllka.velaj@univie.ac.at

Francesco Bonchi
CENTAI, Italy
Eurecat, Spain
bonchi@centai.eu

***Abstract*—Computing basic network properties and machine learning (ML) model outputs, e.g., reachability, shortest path distance, triangle count, node classification, etc., are key to understand large and complex graphs. We study two fundamental problems: (1) Given a graph with uncertain edges and a real-valued network property or an ML model, estimate the uncertainty associated with evaluating the property or the ML model's output over the uncertain graph. (2) Given a limited budget on the number of edges, find the k-best edges whose probability update will reduce the aforementioned uncertainty maximally. We formulate both problems using the information-theoretic notion of entropy and then characterize the hardness of our problems. We next devise approximate solutions with theoretical soundness and greedy subgraph selection-based efficient algorithms. Our empirical evaluation and case study with real-world and synthetic datasets demonstrate that the proposed solutions are more effective and efficient than baselines and are several orders of magnitude faster than exact approaches.**

***Index Terms*—Uncertain graphs, Information entropy, Graph uncertainty estimation, Edge cleaning under budget**

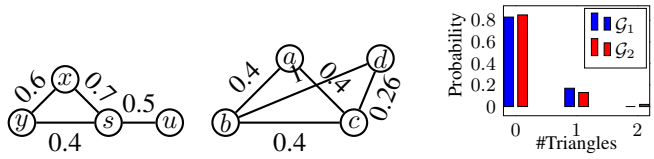
I. Introduction

An uncertain graph is an important data model for real-world networks, where one assigns an independent probability of existence on every edge [1], [2]. An uncertain graph $\mathcal{G}$ is interpreted as a probability distribution over $2^{|E|}$ deterministic graphs (possible worlds), where $|E|$ is the number of edges in $\mathcal{G}$. Uncertainty may arise due to, e.g., measurement errors, data integration, prediction models, injecting uncertainty for obfuscation, and lack of precise information needs. Uncertain graphs have prompted a great deal of research because of their expressiveness in a wide range of applications [3].

The classic approach of mining and machine learning (ML) with deterministic networks starts from the exploration of structural properties and ML model outputs over these graphs. They can be global properties or ML models employed on a graph, e.g., number of triangles, diameter, a graph's class label, etc. Others are local properties or ML models on individual nodes and edges, such as reachability between a pair of nodes, a node's centrality and class label, etc.

We are interested in measuring both local and global properties and ML model outcomes on a network; in particular, *a property or an ML model P that can be denoted as a function*

Arafat and Mobaraki are main contributors and co-first authors. Khan and Mobaraki acknowledge support from the Novo Nordisk Foundation grant NNF22OC0072415.



(a) $\mathcal{G}_1$ (b) $\mathcal{G}_2$ (c)

Fig. 1. (a-b) Uncertain graphs $\mathcal{G}_1$ and $\mathcal{G}_2$. An uncertain graph with $|E|$ edges can instantiate any one of $2^{|E|}$ (deterministic) possible worlds. In such possible worlds, $\mathcal{G}_1$ can have 0 or 1 triangle (Δxys), whereas $\mathcal{G}_2$ may contain 0, 1, or 2 triangles (Δabc, Δbcd). This leads to uncertainty associated with counting triangles in an uncertain graph. (c) Probability distribution of the number of triangles in $\mathcal{G}_1$ and $\mathcal{G}_2$.

$P : G \rightarrow \mathbb{R}$, when computed on a deterministic graph G (or, on some of its components, e.g., nodes or edges), produces a deterministic, real value. Examples include reachability and shortest path (SP) distance between a given pair of nodes, number of triangles, nodes classification into labels, etc. Notice that while an ML model typically associates a probability with its prediction, we disregard such probability in this work. For instance in node classification using a fixed, deterministic graph neural network (GNN), one may consider the predicted class label (i.e., an integer between 1 to c for total c output classes) from the last softmax layer as a deterministic, real-valued output. A fixed, deterministic GNN has (1) all factors which determine the inference process, such as layers and model parameters, among others, fixed; and (2) given the same input, the GNN will always produce the same output. With such deterministic, real-valued network functions, we aim at measuring and reducing the uncertainty that arises due to the uncertainty inherent in graph data.

An uncertain graph with $|E|$ edges leads to $2^{|E|}$ possible worlds. Thus, computing a graph property or an ML model's output on an uncertain graph would involve deriving the expectation over its exponential number of possible worlds, which is often intractable [4]–[6]. For instance, the expected number of triangles in the uncertain graph $\mathcal{G}_1$ in Figure 1(a) is: $\sum_{\Omega=\{0,1\}} \Omega \times Pr\big(\Delta(\mathcal{G}_1) = \Omega\big)$ = 0.168. $Pr\big(\Delta(\mathcal{G}_1) = \Omega\big)$ denotes the aggregated probability of those possible worlds in which the number of triangles is exactly Ω. Similarly, one can also verify that the expected number of triangles in the uncertain graph $\mathcal{G}_2$ in Figure 1(b) is: $\sum_{\Omega=\{0,1,2\}} \Omega \times Pr\big(\Delta(\mathcal{G}_2) = \Omega\big)$ = 0.168. Note that in a possible world, $\mathcal{G}_1$

TABLE I
Uncertainty, estimated via $Entropy$, for computing various network functions is different even on the same uncertain graph $\mathcal{G}_1$. We define Sup, $Conf$, and $Entropy$ in §II.

network function	$Sup(P, \mathcal{G}_1)$	$Conf(\Omega, P, \mathcal{G}_1)$	**entropy** $H(\mathbf{\Omega})$
y-u reachability	0, 1	0.67, 0.33	0.91
Triangle count	0, 1	0.83, 0.17	0.65
y-u SP distance	$2, 3, \infty$	0.20, 0.13, 0.67	1.23
$label(u)$	0, 1	0.50, 0.50	1.00

can have 0 or 1 triangle, whereas $\mathcal{G}_2$ may contain 0, 1, or 2 triangles (Figure 1). Clearly, there is uncertainty associated with counting triangles over an uncertain graph.

Is there the same amount of uncertainty on counting the number of triangles over $\mathcal{G}_1$ and $\mathcal{G}_2$? The answer is no, even though the expected number of triangles is the same in both of them. Intuitively, if most of the probability is concentrated on a specific output value, there is less uncertainty in computing a function P over the possible worlds of $\mathcal{G}$. To this end, the first novel problem that we investigate is as follows. *Given a network function $P : G \rightarrow \mathbb{R}$ and an uncertain graph $\mathcal{G}$, estimate the uncertainty of computing P on $\mathcal{G}$.*

We propose a generic measure via entropy (defined in §II) to estimate the uncertainty of computing a network property or an ML model's output on $\mathcal{G}$. The higher the uncertainty, the greater is the entropy [7].

Given an uncertain graph $\mathcal{G}$, is there the same amount of uncertainty on measuring two different network functions P_1 and P_2? The answer is also negative, as depicted in the example below.

Example 1. *Consider the graph $\mathcal{G}_1$ in Figure 1(a) and four real-valued, deterministic functions: reachability between y and u, shortest path (SP) distance between y and u, triangle count, and class label of u given a classifier $\mathcal{C}$. For simplicity, assume that* **(i)** *$\mathcal{C}$ assigns to an unlabelled node the majority label of its neighbors (if at least one neighbor exists) in a possible world, or 0 otherwise, and* **(ii)** *the labels for nodes x, y, s are all known to be 1. We report the uncertainty of measuring these functions in Table I. The uncertainty values are different for different functions, even on the same uncertain graph. For node labelling, $label(u)$ would be the same as $label(s) = 1$ in $1/2$ of the possible worlds and 0 in the rest. Hence, the entropy associated with predicting $label(u)$ is 1.*

This example illustrates two points: (1) The uncertainty associated with different functions on the same uncertain graph is different, and (2) due to the uncertainty associated with edges in the uncertain graph, a node classifier's output will have uncertainty. Hence, it is critical to formally define and estimate the uncertainty of measuring a network function over an uncertain graph.

The second novel problem that we study is as follows.

Given a network function $P : G \rightarrow \mathbb{R}$, an uncertain graph $\mathcal{G}$, an edge probability update operation $\mathcal{U}\big(p(e)\big) : (0, 1) \rightarrow \{0, 1\}$, and a small budget $k > 0$, find at most k uncertain edges in $\mathcal{G}$ such that if one updates the probabilities of these edges based on the update operation $\mathcal{U}$, the uncertainty of estimating P on $\mathcal{G}$ reduces maximally.

TABLE II
The same edge probability update brings different changes in uncertainty of computing various network functions on $\mathcal{G}_1$.

network function	$H(\mathbf{\Omega})$	**updated edge prob.**	**updated** $H(\mathbf{\Omega})$
y-u reachability	0.91	$p(s, u) = 1$	0.93
Triangle count	0.65	$p(s, u) = 1$	0.65
$label(u)$	1.00	$p(s, u) = 1$	0.00
y-u reachability	0.91	$p(s, x) = 0$	0.72
Triangle count	0.65	$p(s, x) = 0$	0
$label(u)$	1.00	$p(s, x) = 0$	1.00

The same edge probability update could bring different changes in uncertainty associated with measuring various network functions. Therefore, it is critical to find the optimal k-edge set for a specific problem instance, since this optimal set might be different based on various network functions and update operations, even on the same uncertain graph.

Example 2. *Consider $\mathcal{G}_1$ in Figure 1(a), we show in Table II that if we update the probability of edge (s, u) to 1, the uncertainty of y-u reachability increases from 0.91 to 0.93, the uncertainty of predicting $label(u)$ reduces from 1 to 0, while the uncertainty of triangle counting does not decrease. In another example, if we set the probability of edge (x, s) to 0, the uncertainty of triangle counting reduces largely, i.e., it decreases from 0.45 to 0, whereas that of y-u reachability drops from 0.91 to only 0.72, and the uncertainty of predicting $label(u)$ does not change at all.*

Real-world applications of the studied problems. Uncertainty permeates graph data. In sensor networks, reachability computation is key to packet broadcasting. However, the connectivity between sensor nodes is estimated using noisy measurements, leading to edges with a probability of existence [8]. Analogously, interactions in protein networks are established through noisy and error-prone experiments, repeated for a limited number of times, resulting in edge probabilities [9]. Motifs counting and node classification are useful in such networks for predicting co-complex memberships. In road networks, where SP distance computation is critical to traffic routing, uncertain graphs arise due to unexpected traffic jams on road segments [5]. In crowdsourced entity resolution (ER), crowd taskers are asked if two records belong to the same entity or not, leading to an edge between records in a pair that must be grouped together. Reachability computation over this graph structure assists in inferring matching/ non-matching relationship via transitivity and anti-transitivity, thus reducing the cost of crowdsourced ER [10]. However, human workers make mistakes, hence the same record pairs are crowdsourced a limited number of times; and an edge probability between two records is assigned denoting the ratio of crowd workers who answered 'YES' on the question if the two records are the same entity. When the graph structure itself is uncertain as stated, quantifying the impact of uncertainty in the measurement of network properties and ML model outputs naturally becomes important [11]. Uncertainty estimation is key to

benchmark the performance of ML models in downstream tasks, as well as to attest to the reliability and precision of graph property measurement results.

Likewise, reducing the uncertainty associated with evaluating a network property or an ML model's output is important to obtain results with high precision. To this end, we consider two kinds of edge probability updates. The first one increases the probability of an existing uncertain edge to 1 (e.g., enhance the reliability of a link in a sensor network by using a high-quality cable). The second one updates the probability of an existing uncertain edge to either 1 or 0, based on crowdsourcing, additional experiments, and thereby deciding whether that edge exists or not. Setting a small budget for the number of edge updates is required due to cost-effective planning and physical constraints. For instance, crowdsourcing for an uncertain edge, or improving the reliability of a sensor network link requires additional resources such as crowd workers, time, and money.

Challenges and our contributions. There are several technical difficulties in our problems as follows.

Hardness. Uncertainty estimation and reduction of network properties are hard since, in general, computing the underlying properties (such as reliability, shortest path, triangle count, etc.) over uncertain graphs themselves are #P-hard [1], [5], [6]. Hence, approximate solutions with theoretical soundness are desirable. On the other hand, the objective function for uncertainty reduction is not monotonic, neither submodular, nor supermodular (§II), and an exact approach for selecting the k-best edges has exponential time complexity.

Generality and adaptability. Existing methods for uncertainty reduction work in a limited setting, e.g., for reliability query and crowdsourcing-based edge cleaning [12], [13]. They ignore other graph properties, ML model outputs, and diverse kinds of edge probability updates. Heuristics designed therein to work on Boolean-valued graph properties, such as reachability, do not work with arbitrary real-valued graph properties and ML models, e.g., shortest path distance, node classification, and number of triangles (§VI). Moreover, we have demonstrated that ranking and selecting individual edges, followed by [12], [13], lead to sub-optimal solutions for entropy reduction (§V). Finally, it is unclear how efficient sampling and indexing for graph property estimation, e.g., ProbTree indexing [14], recursive stratified sampling (RSS) [15], approximate triangle counting [16], can be employed with those existing solutions.

Our key contributions are summarized below.

- We show that our problems are **NP**-hard, non-monotonic, non-submodular, and non-supermodular under various network functions (§III).
- Despite hardness, the uncertainty estimation algorithm proposed in this paper is generic, i.e., works with any real-valued network property and fixed, deterministic ML models, and comes with an (ϵ, δ)-type guarantee (§IV).
- Our uncertainty reduction algorithm follows a greedy paradigm. We design a practical greedy subgraph-selection strategy to reduce the cold start problem of greedy, while also improving efficiency. We adapt our uncertainty reduction algorithm for two different edge probability update scenarios (§V).
- Experiments show that our algorithms are of high-quality and several orders of magnitude faster than exact methods, while also being more effective and efficient than baselines. Existing sampling (e.g., RSS) and indexing approaches (e.g., probTree) can be coupled with our algorithms to speed up computation. Our case study demonstrates the usefulness of uncertainty reduction in the *strategic collaboration* problem via node classification over uncertain graphs (§VII).

II. Preliminaries

A. Uncertain Graph

Let $\mathcal{G} = (V, E, p)$ be an uncertain graph, where V denotes the set of nodes, $E \subseteq V \times V$ the set of edges, and $p : E \rightarrow (0, 1]$ is a function that assigns a probability of existence to each edge. For simplicity, we consider unweighted, undirected graphs, and the nodes and edges do not have any attributes; however, our solution is applicable to weighted, directed, and attributed graphs. Following the bulk of the literature [1]–[3], we adopt the well-established notion of possible worlds, and assume that edge probabilities are independent of each other: The uncertain graph $\mathcal{G}$ results in a probability distribution over $2^{|E|}$ deterministic graphs (possible worlds). The probability of observing the possible world $G = (V, E_G, W) \sqsubseteq \mathcal{G}$ is:

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

We consider network properties and machine learning models P that can be regarded as functions $P : G \rightarrow \mathbb{R}$, i.e., when evaluated on a deterministic graph G (or, on some of its components, e.g., nodes or edges), produces a deterministic, real value. As run-through examples, we focus on one important global property (triangle count) and two local properties (reachability and the shortest path distance between a given pair of nodes) [4]–[6]. As a concrete use case, we shall discuss node classification in §VII, since a fixed, deterministic node classifier is a function that acts on nodes and their neighbors to generate deterministic, integer-valued labels.

B. Problem Formulation

Consider a network function $P : G \rightarrow \mathbb{R}$ over possible worlds of an uncertain graph $\mathcal{G}$. We define the support set of P on $\mathcal{G}$ as the finite set of real values $P(G)$ for all possible worlds $G \sqsubseteq \mathcal{G}$. Formally,

$$Sup(P, \mathcal{G}) = \{P(G) : G \sqsubseteq \mathcal{G}\} \tag{2}$$

For a specific support value $\Omega \in Sup(P, \mathcal{G})$, we define its confidence as the sum of probabilities of those possible worlds $G \sqsubseteq \mathcal{G}$ in which $P(G) = \Omega$, that is,

$$Conf(\Omega, P, \mathcal{G}) = \sum_{G \sqsubseteq \mathcal{G}} [I(P(G) = \Omega) \times Pr(G)] \tag{3}$$

$I(P(G) = \Omega)$ is an indicator function taking the value 1 if $P(G) = \Omega$, and 0 otherwise. Generalizing Equation 3 to

an arbitrary subset $S \subseteq Sup(P, \mathcal{G})$ and with slight abuse of notation, we define:

$$Conf(S, P, \mathcal{G}) = \sum_{G \sqsubseteq \mathcal{G}} [I(P(G) \in S) \times Pr(G)] \quad (4)$$

$I(P(G) \in S)$ is an indicator function taking the value 1 if $P(G) \in S$, and 0 if $P(G) \notin S$. Notice that $\big(Sup(P, \mathcal{G}), 2^{Sup(P,\mathcal{G})}, Conf(.)\big)$ is a probability space satisfying the Kolmogorov axioms [17]:

1) $0 \leq Conf(\Omega, P, \mathcal{G}) \leq 1$,
2) $Conf(Sup(P, \mathcal{G}), P, \mathcal{G}) = 1$, and
3) for disjoint subsets $\{S_1, S_2, \ldots\}$ of the support set $Sup(P, \mathcal{G})$, $Conf(\cup_{i=1}^{\infty} S_i, P, \mathcal{G}) = \sum_{i=1}^{\infty} Conf(S_i, P, \mathcal{G})$.

We define a random variable $\mathbf{\Omega}$: Different values of $\mathbf{\Omega}$ come from the finite set $Sup(P, \mathcal{G})$. The probability $Pr(\mathbf{\Omega} = \Omega)$ of observing a specific value $\Omega \in Sup(P, \mathcal{G})$ is $Conf(\Omega, P, \mathcal{G})$. We use the following interchangeably: $Pr(\Omega) = Pr(\mathbf{\Omega} = \Omega) = Conf(\Omega, P, \mathcal{G})$.

Intuitively, if most of the probability mass is concentrated on a specific support value, there is less uncertainty in computing function P over the possible worlds of $\mathcal{G}$. In contrast, when the probability mass is evenly distributed across several support values, the uncertainty in computing P would be higher.

We are now ready to define our problems.

Problem 1 (Measuring uncertainty)**.** *Given an uncertain graph $\mathcal{G}$ and a network function $P : G \to \mathbb{R}$, we define the uncertainty of estimating P on $\mathcal{G}$ as the* entropy *of the probability distribution of the random variable $\mathbf{\Omega}$. Formally,*

$$H(\mathbf{\Omega}) = -\sum_{\Omega \in Sup(P,\mathcal{G})} Conf(\Omega, P, \mathcal{G}) \log Conf(\Omega, P, \mathcal{G}) \quad (5)$$

We use logarithm base 2 (Shannon entropy) in the above.

Why we resort to entropy. It is a classic measure of uncertainty of a random variable [7]: *The higher the uncertainty, the greater the entropy*. In particular, it satisfies the three axioms, developed by Shannon, which we also require for uncertainty estimation. **(1)** For a random variable with uniform probabilities, the entropy is a monotonically increasing function of the number of outcomes for the random variable. More choices (i.e., support values) imply greater uncertainty. **(2)** If one splits an outcome category into subcategories, then the new entropy of the extended system should be the sum of the old system's entropy plus the new entropy of the split category weighted by its probability. The creation of more choices (i.e., support values) increases uncertainty weighted according to the likelihood of the increased choice being relevant. **(3)** The entropy is a continuous function of $Conf(\Omega, P, \mathcal{G})$.

We next introduce our second problem: Maximally reduce the uncertainty by updating the probabilities of a small number of edges in $\mathcal{G}$. For update $\mathcal{U}_1\big(p(e)\big) : (0, 1) \to 1$, the resulting edge probability is known apriori; whereas for update $\mathcal{U}_2\big(p(e)\big) : (0, 1) \to \{0, 1\}$, it could be revealed only after the update (e.g., based on crowdsourcing results). We denote by $\mathcal{G}'$ an updated graph of $\mathcal{G}$ after probability updates on the selected edges, and $\mathbf{\Omega'}$ the random variable on $\mathcal{G}'$ in accordance with all $\Omega \in Sup(P, \mathcal{G}')$ and their $Conf(\Omega, P, \mathcal{G}')$ values.

Problem 2 (Reducing uncertainty)**.** *Given an uncertain graph $\mathcal{G} = (V, E, p)$, a network function $P : G \to \mathbb{R}$, an update operation $\mathcal{U}\big(p(e)\big) : (0, 1) \to \{0, 1\}$, and a budget k on the number of edges, select at most k uncertain edges in $\mathcal{G}$ so that on applying the update operation $\mathcal{U}$ on these selected edges, the uncertainty of estimating P on $\mathcal{G}$ reduces maximally, i.e.,*

$$\underset{E_1 \subseteq E, |E_1| \leq k}{\arg\max} \big\{\Delta H(E_1) = H(\mathbf{\Omega}) - H(\mathbf{\Omega'})\big\} \quad (6)$$

Remark. The input uncertain graph $\mathcal{G}$ and the updated uncertain graph $\mathcal{G}'$ have same set of edges, but different probabilities (including some edge probabilities in $\mathcal{G}'$ can be 0 due to update $\mathcal{U}_2$). Thus, for every possible world $G \sqsubseteq \mathcal{G}$, we can find a possible world $G' \sqsubseteq \mathcal{G}'$ so that G and G' are structurally same, and vice versa. However, $Pr(G)$ and $Pr(G')$ may be different due to different edge probabilities in $\mathcal{G}$ and $\mathcal{G}'$. As a consequence, given a function P, the set of support values remains the same over $\mathcal{G}$ and $\mathcal{G}'$; however, the confidence of a specific support value Ω, i.e., $Pr(\Omega)$ can be different in $\mathcal{G}$ and $\mathcal{G}'$ (including $Pr(\Omega)$ may be 0 in one of them).

III. Results on Problems Characterization

Given an uncertain graph $\mathcal{G}$ and a network function $P : G \to \mathbb{R}$, computing the entropy $H(\mathbf{\Omega})$ requires finding every $\Omega \in Sup(P, \mathcal{G})$ and its probability $Conf(\Omega, P, \mathcal{G})$. However, computing the probability that $P(\mathcal{G})$ takes a specific value is often #P-hard. For instance, the probability that the s-t reachability is 1, is the same as computing the s-t reliability, which is #P-hard [1]. The probability that the shortest path distance from s to t is ∞ is the same as computing (1 - s-t reliability), which is also #P-hard. Similarly, the probability that the number of triangles takes a specific value is #P-hard by [6, Theorem 1]. For node classification, consider a classifier that assigns to an unlabelled node u the label that is reachable the maximum number of times from u in its 2-hop, making the computation of node classification entropy #P-hard. Thus, computing the entropy in Problem 1 and reducing the entropy in Problem 2 are difficult. As further evidence of the complexity of Problem 2, we show that its objective function is not monotonic, neither submodular nor supermodular w.r.t. probability updates of more edges.

Lemma 1. *The objective function in Problem 2 is not monotonic with respect to probability updates of more edges.*

We demonstrate non-monotonicity with the examples in Table III. We find that the entropy of computing a network function is not monotonic with respect to both updates $\mathcal{U}_1$ and $\mathcal{U}_2$. Under updates of the same type, entropy may increase or decrease depending on the edges selected for such updates.

Lemma 2. *The objective function in Problem 2 is neither submodular, nor supermodular with respect to probability updates of more edges.*

TABLE III
Non-monotonicity of Problem 2 (uncertainty reduction) w.r.t. probability updates of more edges

network function	$H(\mathbf{\Omega})$	updated edge prob.	updated $H(\mathbf{\Omega})$
$\mathcal{G}_1$: y-u reachability	0.911	$\mathcal{U}_1 : p(s,u) = 1$	0.93 ↑
		$\mathcal{U}_1 : p(s,u) = 1,\ p(s,y) = 1$	0 ↓
		$\mathcal{U}_2 : p(s,x) = 0$	0.72 ↓
		$\mathcal{U}_2 : p(s,x) = 0,\ p(s,y) = 1$	1 ↑
$\mathcal{G}_1$: y-u SP distance	1.225	$\mathcal{U}_1 : p(s,u) = 1$	1.56 ↑
		$\mathcal{U}_1 : p(s,u) = 1,\ p(s,y) = 1$	0 ↓
		$\mathcal{U}_2 : p(s,x) = 0$	0.72 ↓
		$\mathcal{U}_2 : p(s,x) = 0,\ p(s,y) = 1$	1.0 ↑
$\mathcal{G}_2$: Triangle count	0.689	$\mathcal{U}_1 : p(c,d) = 1$	1.22 ↑
		$\mathcal{U}_1 : p(c,d) = 1,\ p(b,c) = 1$	0.63 ↓
		$\mathcal{U}_2 : p(c,d) = 0$	0.34 ↓
		$\mathcal{U}_2 : p(c,d) = 0,\ p(b,c) = 1$	0.63 ↑

We show non-submodularity with the uncertain graph $\mathcal{G}_1$ (Figure 1(a)), triangle count function, update operation $\mathcal{U}_1$, $X : \{p(x,s) = 1\}$, $Y : \{p(x,s) = 1, p(x,y) = 1\}$, $b : p(s,y) = 1$. The entropy of triangle counting after applying updates X and $X \cup \{b\}$ are 0.795 and 0.971, respectively. Thus, the marginal gain (based on reduction in entropy) is: 0.795-0.971=-0.176. Similarly, the entropy of triangle counting after applying updates Y and $Y \cup \{b\}$ are 0.971 and 0.0, respectively, thus the marginal gain is: 0.971-0=0.971. This example shows that the objective function in Problem 2 is not submodular w.r.t. probability updates of more edges.

We show non-supermodularity with an uncertain graph $\mathcal{G}_3$ having same structure as $\mathcal{G}_1$ (Figure 1(a)), but different edge probabilities: $p(x,y)$= 0.3, $p(s,x)$=0.9, $p(s,y)$=0.05, $p(s,u)$=0.7. Consider triangle count function, update operation $\mathcal{U}_1$, $X : \{p(s,u) = 1\}$, $Y : \{p(x,y) = 1, p(s,u) = 1\}$, $b : p(x,s) = 1$. The entropy of triangle counting after applying updates X and $X \cup \{b\}$ are 0.1032 and 0.1124, respectively. Thus, the marginal gain (based on reduction in entropy) is: 0.1032-0.1124 = -0.0092. Analogously, the entropy of triangle counting after applying updates Y and $Y \cup \{b\}$ are 0.265 and 0.286, respectively; thus the marginal gain is: 0.265-0.286 = -0.021. This example shows that Problem 2's objective function is not supermodular w.r.t. probability updates of more edges.

Similarly, one can show non-submodularity and non-supermodularity for other network functions and update operation $\mathcal{U}_2$. We omit them due to interest of space.

IV. Algorithms for Uncertainty Estimation

Given an uncertain graph $\mathcal{G} = (V, E, p)$ and a function $P : G \to \mathbb{R}$, Equation 5 defines the uncertainty of computing P on $\mathcal{G}$. An exact computation of $H(\mathbf{\Omega})$ requires enumerating all possible worlds of $\mathcal{G}$ and computing $Pr(\mathbf{\Omega} = \Omega)$ for every $\Omega \in Sup(P, \mathcal{G})$. This exact method is exponential in the number of edges $|E|$, having time complexity: $\mathcal{O}\big(2^{|E|} \cdot C(P) + |Sup(P, \mathcal{G})|\big)$, where $C(P)$ is the maximum time required by the employed algorithm to evaluate $P(G)$ on a possible world $G \sqsubseteq \mathcal{G}$. It is not feasible to run the exact approach even for a medium-size graph because of combinatorial complexity. Thus, we resort to sampling based efficient estimation of $H[\mathbf{\Omega}]$ (§IV-A), together with an accuracy guarantee (§IV-B).

Algorithm 1 Estimating $Pr(\mathbf{\Omega} = \Omega)$ and $Sup(P, \mathcal{G})$

Require: positive integer T, function $P : G \to \mathbb{R}$, uncertain graph $\mathcal{G} = (V, E, p)$
1: initialize $\hat{Pr}(.) \leftarrow 0$, $\hat{Sup}(P, \mathcal{G}) \leftarrow \phi$
2: **for all** $i = 1, 2, \ldots, T$ **do**
3: sample a world $G_i \sqsubseteq \mathcal{G}$ via independent sampling of edges based on their probabilities
4: compute observed function value: $\Omega = P(G_i)$
5: $\hat{Pr}(\Omega) \leftarrow \hat{Pr}(\Omega) + \frac{1}{T}$
6: $\hat{Sup}(P, \mathcal{G}) \leftarrow \hat{Sup}(P, \mathcal{G}) \cup \{\Omega\}$
7: **return** $\hat{Pr}$, $\hat{Sup}(P, \mathcal{G})$

Algorithm 2 Estimating $H(\mathbf{\Omega})$

Require: positive integers N, T, function $P : G \to \mathbb{R}$, uncertain graph $\mathcal{G} = (V, E, p)$
1: **for all** $i = 1, 2, \ldots, N$ **do**
2: compute sample distribution: $\hat{Pr}_i, \hat{Sup}_i(P, \mathcal{G}) \leftarrow$ Algorithm 1$(T, P, \mathcal{G})$
3: compute sample distribution's entropy: $\hat{H}_i \leftarrow -\sum_{\Omega \in \hat{Sup}_i(P,\mathcal{G})} \hat{Pr}_i(\Omega) \log \hat{Pr}_i(\Omega)$
4: **return** $\frac{1}{N} \sum_{i=1}^{N} \hat{H}_i$

A. Approximate Estimation of Uncertainty

Estimating $Pr(\mathbf{\Omega} = \Omega)$. We approximate the probability $Pr(\mathbf{\Omega} = \Omega)$ via MC-sampling (Algorithm 1).

$$\hat{Pr}(\mathbf{\Omega} = \Omega) = \frac{\sum_{i=1}^{T} I(P(G_i) = \Omega)}{T} \tag{7}$$

where $\{G_1, G_2, \ldots, G_T\}$ are T possible worlds sampled via independent sampling of edges as per their probabilities.

Theorem 1. *$\hat{Pr}(.)$ is an unbiased estimator of $Pr(.)$.*

Proof.

$$E[\hat{Pr}(\mathbf{\Omega} = \Omega)] = \frac{\sum_{i=1}^{T} E\big[I\big(P(G_i) = \Omega\big)\big]}{T}$$
$$= \frac{\sum_{i=1}^{T} \sum_{G \sqsubseteq \mathcal{G}} \big[I\big(P(G) = \Omega\big) \times Pr(G)\big]}{T}$$
$$= \frac{\sum_{i=1}^{T} Pr(\mathbf{\Omega} = \Omega)}{T} = Pr(\mathbf{\Omega} = \Omega)$$

The last equality follows because $Pr(\mathbf{\Omega} = \Omega) = Conf(\Omega, P, \mathcal{G})$ is constant for given P and $\mathcal{G}$, irrespective of the possible world under consideration. □

Estimating $H[\mathbf{\Omega}]$. As an estimate for $H(\mathbf{\Omega})$, we use the entropy of the sample distribution computed in Algorithm 1.

$$\hat{H} = -\sum_{\Omega \in \hat{Sup}(P,\mathcal{G})} \hat{Pr}(\mathbf{\Omega} = \Omega) \log \hat{Pr}(\mathbf{\Omega} = \Omega) \tag{8}$$

Given a user-specified integer $N > 0$, Algorithm 2 runs Algorithm 1 N times to compute N independent entropy estimates $\hat{H}_1, \hat{H}_2, \ldots, \hat{H}_N$, and returns the average of those N estimates. Computing the average of those N entropy estimates will later help us derive a guarantee on the deviation of the output of Algorithm 2 from the true value $H(\mathbf{\Omega})$.

Time complexity. The time complexity of our entropy estimation method (Algorithm 2) is $\mathcal{O}\big(N \cdot T \cdot C(P)\big)$, where $C(P)$ is the maximum time required by the employed algorithm to evaluate $P(G)$ on a possible world $G \sqsubseteq \mathcal{G}$.

B. Accuracy Guarantee

Algorithm 2 has entropy estimation error $\left|\frac{1}{N}\sum \hat{H}_i - H(\mathbf{\Omega})\right|$ with a probabilistic bound (Theorem 2).

Theorem 2. *For all $N, T > 0$ (input to Algorithm 2) and $\epsilon > 0$,*

$$Pr\left(\left|\frac{\sum \hat{H}_i}{N} - H(\mathbf{\Omega})\right| \geq \frac{|Sup(P,\mathcal{G})| - 1}{2T} + \epsilon\right) \leq 2e^{\frac{-2N\epsilon^2}{\log^2|Sup(P,\mathcal{G})|}} \tag{9}$$

The proof is given in Appendix IX.

Practical effectiveness of our bounds. Our uncertainty estimation method is simple – it only requires a network function computation over $N \cdot T$ possible worlds, yet this comes with an accuracy guarantee, and is generic for any network function $P : G \to \mathbb{R}$. In Theorem 2, we refer to $\frac{|Sup(P,\mathcal{G})|-1}{2T} + \epsilon$ as the margin of error. We observe that larger T decreases the margin of error. In contrast, N has little impact on the margin of error; however, the probability that our error estimate crosses that margin increases as we reduce N. When the support size $|Sup(P,\mathcal{G})|$ increases, both the margin of error and the probability that our error estimate crosses that margin increase. Therefore, a combination of P and $\mathcal{G}$ that results in lower support size $|Sup(P,\mathcal{G})|$ is preferred.

V. Algorithms for Uncertainty Reduction

Given a small budget $k > 0$ on the number of edges, we investigate the problem of selecting up to k uncertain edges, on updating whose probabilities via an operation $\mathcal{U}(p(e))$, the entropy $H[\mathbf{\Omega}]$ is maximally reduced. Our objective function is not monotonic with respect to probability updates of more edges (Lemma 1). Hence, we may not exhaust the full budget k to achieve the maximum reduction in entropy. For simplicity, we consider the 'known' update outcome with $\mathcal{U}_1 : (0,1) \to 1$ in §V-A-V-B. The 'unknown' update outcome for $\mathcal{U}_2 : (0,1) \to \{0,1\}$ is discussed in §V-C.

A. Naïve Approach

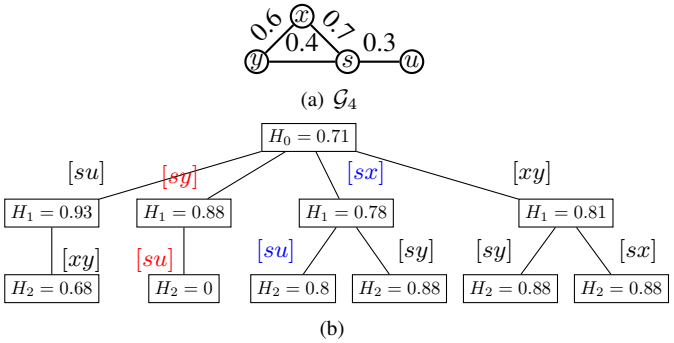
A naïve but exact approach is to enumerate all $\binom{|E|}{i}$ i-size subsets, $0 \leq i \leq k$, $S_i \subset E$; create a new uncertain graph $\mathcal{G}'$ associated to every choice S_i, such that edges in $E \setminus S_i$ retain their original probability, and edges in S_i have updated probabilities in $\mathcal{G}'$ via the update operation $\mathcal{U}_1$. Finally, we compute the updated entropy and select the subset whose update reduces the initial entropy maximally.

The entropy must be recomputed for every choice of S_i and computing entropy exactly over an uncertain graph with $|E|$ edges costs $\mathcal{O}\big(2^{|E|}C(P)\big)$. Thus, the time complexity of the naïve approach is: $\mathcal{O}\big(\sum_{i=0}^{k}\binom{|E|}{i} \cdot 2^{|E|} \cdot C(P)\big)$. Here, $C(P)$ denotes the maximum time required to evaluate function $P(G)$ on a possible world $G \sqsubseteq \mathcal{G}$. For a large-scale graph, such an exact approach is impractical.

Algorithm 3 `Greedy` Edge Selection with Sampling

Require: uncertain graph $\mathcal{G} = (V, E, p)$, function P, budget k on # edge updates

1: $E_0^* \leftarrow \emptyset$
2: **for** $i = 1$ **to** k **do**
3: $\quad e^* \leftarrow \arg\max_{e \in E \setminus E_{i-1}^*} \left\{\Delta\hat{H}(E_{i-1}^* \cup \{e\})\right\}$
4: $\quad E_i^* \leftarrow E_{i-1}^* \cup \{e^*\}$
5: $\quad p(e^*) \leftarrow \mathcal{U}_1\big(p(e^*)\big) = 1$
6: **return** $\arg\max_{E_i^*, 0 \leq i \leq k} \left\{\Delta\hat{H}(E_i^*)\right\}$



(a) $\mathcal{G}_4$

(b)

Fig. 2. Reducing entropy of y-u reachability on uncertain graph $\mathcal{G}_4$: The edges that should be chosen for cleaning as per $\mathcal{U}_1$ to reach global optima are colored red ($[sy], [su]$). The edges chosen by Greedy are colored blue. Greedy selects the edge $[sx]$ at round 1 because it is locally best at round 1. However, this leads to a globally sub-optimal selection ($[sx], [su]$).

B. Greedy Algorithm with Sampling

We improve the efficiency of the naïve approach in two ways: **(1)** We eliminate the exponential number of subset choices by exploiting linear-time, greedy edge-selection, which reduces the time complexity term from $\sum_{i=0}^{k}\binom{|E|}{i}$ to $\mathcal{O}(k|E|)$. **(2)** We employ MC-sampling to approximate entropy values (§IV-A). It reduces the exponential term $2^{|E|}$ to a much smaller term $N \cdot T$ at the cost of a predefined margin of error. Our greedy heuristic with sampling is given in Algorithm 3. At every iteration i (Lines 2-5), we greedily find the optimal edge set E_i^* that reduces the entropy maximally, compared to any other edge set having the same cardinality. Among these E_i^*'s, $0 \leq i \leq k$, we return the one resulting in the maximum decrease in entropy (Line 6). Furthermore, instead of computing entropy exactly, we leverage Algorithm 2 (§IV-A) to estimate it via MC-sampling (Line 3), resulting in overall time complexity of Algorithm 3 to be $\mathcal{O}(k \cdot |E| \cdot N \cdot T \cdot C(P))$. Recall that the underlying objective function is non-submodular (Lemma 2). Hence, the greedy algorithm does not provide any approximation guarantee.

Drawbacks. The greedy edge selection with sampling approach (Algorithm 3) suffers from both efficiency and effectiveness issues. **(1)** Efficiency-wise at each iteration, **Greedy** computes the goodness of every remaining edge via MC-sampling (Line 3). However, many edges are less relevant w.r.t. a certain downstream task. For instance, edges that are far away from a specific source-target pair would generally have lower impacts on the reachability or shortest path distance functions between that source-target pair. Thus,

verifying all remaining edges at every iteration is redundant. **(2)** Effectiveness-wise, **Greedy** suffers from the so-called cold start problem, where there are very few local choices that lead to global optima. At earlier rounds, there could be several edges, updating of whose probabilities will lead to quite small (or, even negative) entropy reduction. A locally-best solution at earlier rounds may lead to a globally sub-optimal solution, which is illustrated via an example in Figure 2.

C. Greedy Subgraphs Selection

Since the greedy algorithm cannot know the global outcomes of its local choices at initial rounds, ranking and selecting individual edges based on entropy leads to sub-optimal solutions. Instead, it is desirable to consider *all* relevant edges holistically (w.r.t. the network function), for selecting up to the best k-edges. We propose a more practical algorithm **Greedy+subgraph**. It ranks "subgraphs" of interest based on the network function, update operation, and the entropy of subgraphs. Then, it selects the best subgraph and cleans all its edges together following the update operation. The algorithm continues with the next best subgraph until the budget on the number of edges is exhausted. We first illustrate the key idea for reachability and SP distance functions, with update $\mathcal{U}_1$.

Greedy+subgraph algorithm for reachability and shortest path (SP) distance functions, with update $\mathcal{U}_1$. Here, the "subgraph" of interest is the shortest path between the given s-t pair. In particular, we compute the shortest path between the s-t pair, without considering edge probabilities in $\mathcal{G}$. When there are multiple shortest paths between the s-t pair, we pick any one of them. If the path has no more edges than the budget constraint, all edges on this path are updated to have probability 1 due to $\mathcal{U}_1$. This reduces the entropy to 0, resulting in the maximum possible reduction in entropy. Otherwise, if the path has more edges than the budget constraint k, we select k edges on this path with the highest individual entropy and update their probabilities to 1 in order. Let E_i^* denote the selected edge set at round i, $0 \leq i \leq k$. Among these E_i^*'s, we return the one resulting in the maximum decrease in entropy. The time complexity is dominated by the shortest path computation, which is $\mathcal{O}(|V| + |E| \log |V|)$ via Dijkstra.

Example 3. *Consider budget k = 2. To reduce the entropy of y-u reachability on the uncertain graph $\mathcal{G}_4$ in Figure 2, we compute the shortest path from y to u in $\mathcal{G}_4$ (without considering probability on edges): $y - s - u$. We increase the probability of both edges (ys and su) on this path to 1 via the update operation $\mathcal{U}_1$. This reduces the entropy of y-u reachability on $\mathcal{G}_4$ to 0. Hence, the entropy reduces maximally, and we avoid the cold start problem due to **Greedy**.*

Generalizing "subgraphs" of interest for other network functions. Intuitively, a subgraph of interest constitutes a set of edges so that collectively knowing their existence significantly impacts the outcome of the network function. In this work, we exploit technical knowledge about the network function to select its subgraphs of interest, e.g., the shortest path between the s-t pair for reachability and SP distance functions. For triangle count, we use the triangles in $\mathcal{G}$ as subgraphs of interest. For node classification, we consider the explanation subgraphs [18] in a node's neighborhood that can explain the prediction of a graph neural network. We keep it an open problem to *automatically* derive the characteristics of interesting subgraphs independent of different network functions.

Algorithm 4 Greedy+subgraph Algorithm

Require: uncertain graph $\mathcal{G} = (V, E, p)$, function P, budget k, hyper-parameter r
1: compute the top-r subgraphs $\mathbf{S}$
2: sort subgraphs $S \in \mathbf{S}$ based on entropy
3: $E_0^* \leftarrow \emptyset$, $i \leftarrow 0$
4: **while** $\mathbf{S}$ is not empty $\wedge$ $|E_i^*| < k$ **do**
5: $S^* \leftarrow pop(\mathbf{S})$
6: $i \leftarrow i + 1$
7: $E_i^* \leftarrow E_{i-1}^* \cup Edges(S^*)$
8: clean all edges in S^* via update operation
9: update entropy of remaining subgraphs having common edge with S^*; and re-sort $\mathbf{S}$ based on updated entropy
10: $r' \leftarrow \arg\max_i |E_i^*| \leq k$
11: **return** $\arg\max_{E_i^*, 0 \leq i \leq r'} \left\{ \Delta \hat{H}(E_i^*) \right\}$

Greedy+subgraph for other network functions, with update $\mathcal{U}_1$ or $\mathcal{U}_2$. For a given network function, we select the top-r subgraphs of interest. Here, r is a hyper-parameter that could be as small as 1 for certain network functions and update operation (e.g., as stated, for reachability and SP distance functions, together with update $\mathcal{U}_1$), but can also be larger otherwise (e.g., for update $\mathcal{U}_2$), and is decided empirically. Generally, the total number of edges in these r subgraphs should be $\geq k$, the budget constraint. These provisions for checking up to the budget constraint. Next, we consider the subgraphs one at a time in descending order of entropy, where the entropy of a subgraph S is: $H(S) = -Pr(S) \log Pr(S) - (1 - Pr(S)) \log(1 - Pr(S))$, with $Pr(S) = \Pi_{e \in S} p(e)$. For each subgraph, we clean all its edges via the update operation, until the budget on the number of edges is exhausted. After cleaning a subgraph, we recompute the entropy of those subgraphs that share an edge with the cleaned subgraph and update the sorted order of remaining subgraphs based on their new entropy. Due to the budget constraint, assume that we can clean $r' \leq r$ subgraphs. Let E_i^* denote the selected edge set at round i, $0 \leq i \leq r'$. Among these E_i^*'s, we return the one resulting in the maximum decrease in entropy. The complete procedure is given in Algorithm 4.

Remark. For reachability and SP distance functions, the top-r shortest paths in $\mathcal{G}$ are found using the well-known Eppstein's algorithm (without considering probability on edges). For triangle count, we aim at selecting the top-r triangles with the highest entropy. We first sort the edges in descending order of their individual entropy, iterate over the sorted edges, and check if the endpoints create a triangle. We sample r triangles in this manner. For node classification, the selection of the top-r explanation subgraphs via [18] is discussed in §VII.

VI. Related Work

Edge cleaning to reduce uncertainty. Lin et al. [12] studied crowdsourcing based edge cleaning (i.e., only update operation $\mathcal{U}_2$) to improve the quality of distance-constrained s-t reachability queries over uncertain graphs. The quality is measured via entropy, and the objective is to select the k-best edges that result in the maximum expected quality improvement for a given set of distance-constrained s-t reachability queries. Wu et al. [13] additionally included crowd noise in the above problem. While the essence of these problems is similar to that of ours, albeit [12], [13] focused only on Boolean-valued graph properties, such as distance-constrained reachability. Specifically, the heuristic techniques designed therein (e.g., all length-d paths enumeration from s to t, recursive computation, edge pruning optimizations, etc.) to work on Boolean-valued graph properties do not generalize with arbitrary real-valued graph functions, such as SP distance, triangle count, and node classification that we consider in this work. Additionally, we have shown that ranking and selection of individual edges lead to sub-optimal solutions for entropy reduction (§V, §VII). Besides, it is unclear how efficient sampling and indexing for network property estimation, e.g., ProbTree indexing [14], recursive stratified sampling [15], approximate triangle counting [16], can be employed with those existing solutions (§VII).

Uncertainty in graph ML. In graph neural networks, uncertainty typically arises from two sources: **(1)** data or statistical uncertainty (*aleatoric uncertainty*), due to data containing noise and error, or lacking sufficient information, including edge uncertainty, and **(2)** systemic or model uncertainty (*epistemic uncertainty*), due to not knowing the distribution of true labels, choice of model structures, and selection of parameters. For surveys, we refer to [11], [19]. We focus on the first type of uncertainty – reduction of which received little attention in the past. Node classification in uncertain graphs has been studied in [20], [21] by following Bayesian and embedding methods, respectively. To the best of our knowledge, ours is the first work that employs the widely studied possible world semantics to estimate and reduce the uncertainty associated with fixed, deterministic node classifiers over uncertain graphs.

VII. Experimental Results

Experimental setup. We experiment using the single core of a server with 128 AMD EPYC processor and 256GB RAM. Our codebase is at **https://github.com/toggled/uncertainty**.

We conduct experiments with five real-world and one synthetic datasets (Table IV): **(1)** *ER* is a synthetic graph according to the Erdős-Rényi's random graph model. Edge probabilities are generated by sampling from a Gaussian distribution with mean 0.27 and standard deviation 0.21, so that the edge probability distribution closely follows that of the real-world dataset *Biomine*. **(2)** *Biomine* graph [22] is built by integrating cross-references from various biological databases. Nodes denote biological entities and edges represent real-world phenomena between two nodes. Edge probabilities are derived in [22] based on three criteria: relevance, informativeness, and confidence in the existence of a relationship.

TABLE IV
Statistics of datasets. Reach: reachability, SP: shortest path distance, #Tri: triangle counting, NC: node classification

graph	type	queries shown	#nodes	#edges	edge prob.
ER	synthetic	reach, SP, #Tri	15	22	0.27 ± 0.21
Biomine	biological	reach, SP	1 008 201	13 485 878	0.27 ± 0.21
Flickr	social	#Tri	78 322	10 171 509	0.09 ± 0.06
Products	crowdsourced	reach	2 173	37 641	0.17 ± 0.09
Papers	crowdsourced	#Tri	995	152 731	0.26 ± 0.23
DBLP	collaboration	NC	632 870	3 301 970	0.46 ± 0.14

TABLE V
Entropy estimate: comparison with **Exact** method (*ER*)

algorithm	avg. running time (sec)			avg. error		
	Reach	SP	#Tri	Reach	SP	#Tri
Exact	177.4	190.9	815.8	0	0	0
MC	0.039	0.096	0.368	0.088	0.086	0.058

(3) *Flickr* is an online social network, where users share photos and join common interest groups. We obtain a subset of the network from [2], where the probability of the edge between any two users is computed as the Jaccard coefficient of the interest groups that the two users belong to. **(4)** *Products* is a popular crowdsourcing-based entity resolution dataset [10], with entities (i.e., nodes) being products. **(5)** *Papers* is also an entity resolution dataset [10], where research papers are entities. About 1.5% and 30% of randomly selected entity pairs (i.e., edges) from these datasets, respectively, are crowdsourced by Amazon's Mechanical Turk platform. Five workers per task are engaged. Each task asks if the entities in a pair (i.e., end-nodes of an edge) are the same or not. The edge probability is assigned as the number of workers who voted 'yes', divided by the total number of workers being asked for that pair. **(6)** *DBLP* is a collaboration network widely used in the literature [23], [24]. We downloaded it on Sep. 1, 2023. Each node is an author and edges denote co-author relations. For every edge (u, v), we compute the number of collaborations c between u and v, and we assign probability $p((u, v)) = 1 - e^{-c/\mu}$, $\mu = 2$ [4].

We vary parameters N and T that control the number of MC-samples (§IV). The details are deferred to Appendix X. We observe that irrespective of network functions, the entropy estimate initially fluctuates as we increase T; and after a certain T (say $T = T^*$), it gets stable indicating that T^* is sufficient to approximate entropy. We determine T^* based on the entropy rate: $\frac{\Delta H}{\Delta T} = \frac{H_{T+\delta} - H_T}{\delta}$. Here, $\delta > 0$ is the increment in T. We set a threshold $\epsilon_T = 0.03$ for all network functions and datasets, and determine T^* as the point after which $|\frac{\Delta H}{\Delta T}| \leq \epsilon_T$. We select optimal N, denoted as N^*, similarly by setting a threshold $\epsilon_N = 0.001$ and computing N^* as the point after which $|\frac{\Delta H}{\Delta N}| \leq \epsilon_N$. Typically, we find that $N^* \approx$ 50-500 and $T^* \approx$ 5-50 for different network functions.

A. Uncertainty Estimation Results

For reachability (Reach) and shortest path distance (SP), we construct query sets Q_2, Q_4, Q_6: Each set containing uniformly at random selected 100 s-t pairs from the input graph that are 2-hops, 4-hops, and 6-hops away, respectively.

Since *ER* is a small graph, its query sets include only Q_2 and Q_4, each set containing 5 s-t pairs. The results are reported as an average over all queries in these query sets. For triangle counting (#Tri), we count the number of triangles in deterministic possible worlds. The number of support values for #Tri queries is extremely large, thus we use uniform bucketing to group nearby support values and estimate entropy.

For uncertainty estimation, our proposed Algorithm 2, referred to as **MC**, leads to several variants based on the downstream network functions (e.g., reachability, SP distance, triangle counting), indexing technique (e.g., ProbTree [14]), and advanced sampling strategy (e.g., RSS [15]). For instance, **MC+BFS** evaluates reachability by running probabilistic BFS [4], **MC+Dijkstra** evaluates SP distance by running probabilistic Dijkstra [5], and **MC+ApproxTriangle** evaluates triangle count via MC-sampling of triplets [16]. Further adopting ProbTree indexing for reachability and SP distance evaluation leads to **ProbTree-MC+BFS** and **ProbTree-MC+Dijkstra**. Similarly, instead of using MC-sampling to estimate entropy, adopting RSS leads to algorithms **RSS** and further adoption of ProbTree leads to **ProbTree-RSS**.

1) Comparison w.r.t. to an exact method: Table V shows efficiency comparison and approximation error of **MC** w.r.t. the **Exact** algorithm for uncertainty estimation (§IV). **MC** is 3 orders of magnitude faster than **Exact**, while incurring small errors in entropy estimation. As **Exact** is impractical on larger or even medium-size graphs due to its computational complexity, our results show that **MC** and its variants can be used as effective proxy over larger datasets.

2) Efficiency, effectiveness, and memory usage on larger datasets: Tables VI, VII, and VIII show performance comparison of the algorithms on larger *Biomine* and *Flickr* datasets. We consider **MC** as the baseline and compute errors in entropy estimation of other algorithms with respect to it. In Table VI for reachability queries, **(1) MC** is the least efficient, while **ProbTree-RSS** is the most efficient among all the algorithms. The latter benefits from both ProbTree indexing and RSS sampling. In general, ProbTree indexing improves the running time of **MC**, **MC+BFS**, and **RSS** algorithms. Among purely sampling-based methods, **RSS** is the most efficient, followed by **MC+BFS**, whereas **MC** is the least efficient. **(2)** Accuracy-wise, **MC+BFS** is the most similar to **MC**. Moreover, all our algorithms generally have lower average errors w.r.t. **MC**. **(3)** Purely sampling-based algorithms consume the least amount of memory, while ProbTree-based methods require additional storage due to indexing. Similar trends can be observed for SP distance function in Table VII and triangle count in Table VIII.

TABLE VI
Entropy estimate for reachability (*Biomine*)

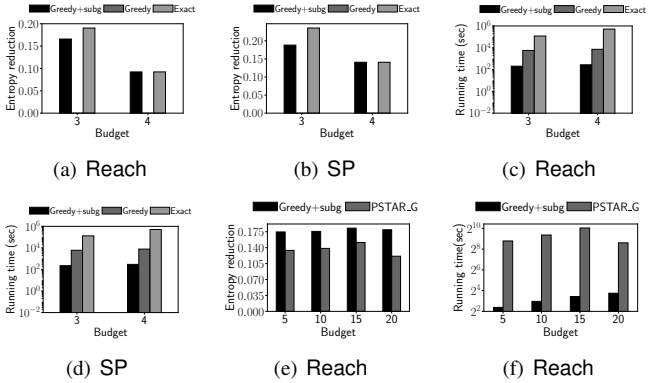
algorithm	avg. running time (sec)	avg. error	avg. peak mem. (GB)
MC	32849.5	**0**	**4.0**
MC+BFS	2742.2	0.005	**4.0**
ProbTree-MC	18515.1	0.008	8.6
ProbTree-MC+BFS	2257.1	0.049	8.6
RSS	1672.1	0.100	**4.0**
ProbTree-RSS	**1342.8**	0.300	8.6

TABLE VII
Entropy estimate for shortest path distance (*Biomine*)

algorithm	avg. running time (sec)	avg. error	avg. peak mem. (GB)
MC	26715.3	**0**	**4.0**
MC+Dijkstra	4220.7	0.033	**4.0**
ProbTree-MC	24559.4	0.038	8.6
ProbTree-MC+Dijkstra	**4154.1**	0.011	8.6

TABLE VIII
Entropy estimate for triangle count (*Flickr*)

algorithm	avg. running time (sec)	avg. error	avg. peak mem. (GB)
MC	313708.9	**0**	3.1
MC+ApproxTriangle	**34860.2**	0.38	**3.0**



(a) Reach (b) SP (c) Reach

(d) SP (e) Reach (f) Reach

Fig. 3. (a)-(d): Comparison among **Exact**, **Greedy**, and **Greedy+subgraph**, *ER* dataset, update $\mathcal{U}_1$. Each s-t pair is 3-hops away when budget = 3, and 4-hops away when budget = 4. **Greedy** often does not reduce entropy at all due to the cold start problem. (e)-(f): Comparison between our **Greedy+subgraph** and existing **PSTAR_G** [12], *products* dataset, update $\mathcal{U}_2$.

B. Uncertainty Reduction Results

We study the effectiveness of **Greedy+subgraph** in uncertainty reduction. The impact of parameters r and k on **Greedy+subgraph** is deferred to the Appendix X.

*1) **Exact**, **Greedy**, and **Greedy+subgraph** comparison:* We compare the effectiveness and efficiency of our ultimately proposed **Greedy+subgraph** for entropy reduction (§V), with **Greedy** and **Exact** algorithms on reachability and SP distance functions. We use the smallest *ER* dataset for these experiments due to the computational cost of running the **Exact** method. Figures 3(a-b) present effectiveness based on the amount of entropy reduced (y-axis) after selecting and cleaning edges as per update $\mathcal{U}_1$ under various budgets (x-axis). Notice that **Greedy+subgraph** reduces almost equal amount of uncertainty as **Exact**, whereas **Greedy** cannot often reduce uncertainty at all due to the cold start problem. Figures 3(c-d) show the efficiency of the algorithms. **Greedy+subgraph** is at least 4-5 orders of magnitude faster than **Exact** and about 2 orders of magnitude more efficient than **Greedy**. These results demonstrate the effectiveness and efficiency of our **Greedy+subgraph** method for entropy reduction.

*2) Comparison with existing **PSTAR_G** [12]:* We employ **PSTAR_G** as a baseline, which was proposed to find a limited number of edges whose cleaning as per $\mathcal{U}_2$ would maximally reduce the expected entropy of d-hop reachability. We compare it with our algorithm **Greedy+subgraph** using 100 query pairs, where each source-target pair is 5-hops away, on the *products* dataset. In Figure 3(e-f), **Greedy+subgraph**

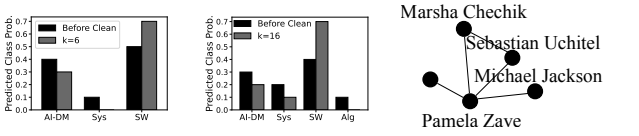
is more effective than **PSTAR_G** in entropy reduction for the same budget, since we consider greedy subgraph selection to remove the cold start problem of greedy methods, whereas **PSTAR_G** follows a greedy approach based on ranking and selection of individual edges. Efficinecy-wise, our algorithm **Greedy+subgraph** achieves 32-128X speedup than **PSTAR_G**. Recall that **PSTAR_G** computes expected entropy reduction for all candidate edges using MC sampling, whereas **Greedy+subgraph** selectively cleans edges along the most uncertain subgraphs that are relevant to the network function.

C. ML Application in Strategic Collaboration Problem

We showcase a concrete application of our problems and their solutions in the *strategic collaboration* problem: Find co-authors to collaborate often such that someone's profile is more prominently classified in a specific research domain. We resolve this problem by estimating and reducing uncertainty of node classification over the uncertain *DBLP* collaboration network (Table IV). A node can have one of the 8 class labels: Artificial Intelligence & Data Mining (AI-DM), Databases (DB), Hardware & Architecture (HW), Media & Applications (Media), System Technology (Sys), Programming Languages & Software Engineering (SW), Algorithms & Theory (Alg), and Biomedical (Bio). The most frequent label in an author's publications from a set of top conferences and journals is used as the author's ground-truth class label. The top-20 frequent and relevant keywords present in the paper titles from each class label are used as binary node features. We randomly selected 80% nodes for training the node classifier and 20% for testing.

We generate 100 possible worlds from the uncertain *DBLP* graph and train a 3-layer vanilla GCN [25] on the labeled nodes from these possible worlds in a supervised manner. For a specific test node, we obtain its predicted class labels across 10 selected possible worlds. Figure 4 shows the distribution of predicted class labels for two well-known researchers: Jean-Pierre Briot (https://dblp.org/pid/b/JPBriot.html) and Pamela Zave (https://dblp.org/pid/z/PamelaZave.html) over these possible worlds. The predicted class labels are relevant: JP Briot and his collaborators published at several AI-DM, Systems, and SW venues, while he mainly published in SW conferences. Pamela Zave is a senior researcher known for her work in *software requirement engineering*; she and her collaborators over the years published in many areas including AI-DM, Sys, SW, and Alg. Using our algorithm (§IV), we estimate the uncertainty associated with label predictions by a GNN [25]; the respective entropies are 1.36 and 1.84 for JP Briot and Pamela Zave over the uncertain *DBLP* graph.

To effectively reduce uncertainty within a budget, our **Greedy+subgraph** algorithm in §V selectively cleans edges (we consider update $\mathcal{U}_1$) along subgraphs of interest that are highly relevant to that node's classification. To this end, we employ GNN explainability methods, such as **SubgraphX** [18] that identifies an important subgraph in a node's neighborhood which explains the GNN's prediction. Specifically for a test node, we find its majority predicted class (e.g., SW for Pamela Zave) and apply **SubgraphX** on each possible world to obtain an explanation subgraph that explains the majority class prediction in that possible world. With 10 possible worlds, we obtain 10 explanation subgraphs which we use as candidates for $\mathbf{S}$ in Algorithm 4 (Line 1). We hypothesize that by cleaning the most uncertain (high-entropy) ones among these important subgraphs, one can reduce the uncertainty of that node's label prediction. Figures 4(a-b) validate our hypothesis: After cleaning the top-16 uncertain edges selected via our **Greedy+subgraph** method from important subgraphs, Pamela Zave is classified into the SW class in more possible worlds, and the corresponding entropy reduces to 1.16.



(a) JP Briot (b) Pamela Zave (c) Recommended collaborations

Fig. 4. (a-b) The distributions of predicted class for Jean-Pierre Briot and Pamela Zave on *DBLP* graph before and after cleaning of selected uncertain edges. (c) The recommended future collaborations for Pamela Zave (among her co-authors) such that she is more prominently classified into SW.

In Figure 4(c), we highlight a few important edges that are selected and cleaned via **Greedy+subgraph**. Further inspecting the cleaned edges, we find that many of them connect Pamela Zave to prominent researchers in the SW domain, e.g., Marsha Chechik (https://dblp.org/pid/c/MarshaChechik.html), Sebastian Uchitel (https://dblp.org/pid/21/1391.html) and Michael A. Jackson (https://dblp.org/pid/92/1629-1.html). By cleaning these edge probabilities to 1, i.e., recommending to collaborate more with these co-authors, Pamela Zave would be predicted to have the SW class label across more possible worlds, thus reducing the uncertainty associated with this author's classification. Our case study demonstrates the usefulness of our problems and their solutions over node classification in real-world uncertain graphs.

VIII. Conclusions

We studied estimating and reducing uncertainty of computing network functions over uncertain graphs. We exploited the information-theoretic notion of entropy to formulate our problems and characterize their complexity. For uncertainty estimation, we proposed an approximation algorithm with an (ϵ, δ)-type guarantee. For uncertainty reduction, we designed a practical greedy subgraphs selection algorithm that reduces the cold start problem of greedy approaches. Based on empirical results, our algorithms coupled with indexing and smart sampling strategies achieve the best accuracy and efficiency. Our case study depicted an application of uncertainty reduction for node classification in the strategic collaboration problem.

One limitation of this work is that we consider a network function that produces a single value when computed on a deterministic graph. In future, we shall extend our solution to network functions generating multiple outputs, e.g., all subgraphs satisfying an input constraint, all nodes reachable within a limited number of hops, all nodes classified in a specific label, etc.

REFERENCES

[1] M. O. Ball, "Computational complexity of network reliability analysis: An overview," *IEEE Tran. Rel.*, 1986.
[2] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, "K-nearest neighbors in uncertain graphs," *PVLDB*, 2010.
[3] A. Khan, Y. Ye, and L. Chen, *On Uncertain Graphs*, ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2018.
[4] X. Ke, A. Khan, and L. L. H. Quan, "An in-depth comparison of s-t reliability algorithms over uncertain graphs," *PVLDB*, 2019.
[5] A. Saha, R. Brokkelkamp, Y. Velaj, A. Khan, and F. Bonchi, "Shortest paths and centrality in uncertain networks," *PVLDB*, 2021.
[6] C. Ma, R. Cheng, L. V. S. Lakshmanan, T. Grubenmann, Y. Fang, and X. Li, "LINC: A motif counting algorithm for uncertain graphs," *PVLDB*, 2019.
[7] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, 1948.
[8] J. Ghosh, H. Q. Ngo, S. Yoon, and C. Qiao, "On a routing problem within probabilistic graphs and its application to intermittently connected networks," in *INFOCOM*, 2007.
[9] S. Asthana, O. D. King, F. D. Gibbons, and F. P. Roth, "Predicting protein complex membership using probabilistic network reliability," *Genome Res.*, 2004.
[10] S. Wang, X. Xiao, and C. Lee, "Crowd-based deduplication: An adaptive approach," in *SIGMOD*, 2015.
[11] F. Wang, Y. Liu, K. Liu, Y. Wang, S. Medya, and P. S. Yu, "Uncertainty in graph neural networks: A survey," *CoRR*, vol. abs/2403.07185, 2024.
[12] X. Lin, Y. Peng, B. Choi, and J. Xu, "Human-powered data cleaning for probabilistic reachability queries on uncertain graphs," *TKDE*, 2017.
[13] Y. Wu, X. Lin, Y. Yang, and L. He, "Cleaning uncertain graphs via noisy crowdsourcing," *World Wide Web*, 2019.
[14] S. Maniu, R. Cheng, and P. Senellart, "An indexing framework for queries on probabilistic graphs," *ACM Trans. Database Syst.*, 2017.
[15] R. Li, J. X. Yu, R. Mao, and T. Jin, "Recursive stratified sampling: A new framework for query evaluation on uncertain graphs," *TKDE*, 2016.
[16] T. Schank and D. Wagner, "Approximating clustering coefficient and transitivity," *J. Graph Alg. Appl.*, 2005.
[17] A. N. Kolmogorov, "Foundations of the theory of probability," 1960.
[18] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *ICML*, 2021.
[19] S. Munikoti, D. Agarwal, L. Das, and B. Natarajan, "A general framework for quantifying aleatoric and epistemic uncertainty in graph neural networks," *Neurocomputing*, 2023.
[20] M. Dallachiesa, C. C. Aggarwal, and T. Palpanas, "Node classification in uncertain graphs," in *SSDBM*, 2014.
[21] J. Hu, R. Cheng, Z. Huang, Y. Fang, and S. Luo, "On embedding uncertain graphs," in *CIKM*, 2017.
[22] L. Eronen and H. Toivonen, "Biomine: Predicting links between biological entities using network models of heterogeneous databases," *BMC Bioinformatics*, 2012.
[23] M. Ley, "Dblp: some lessons learned," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1493–1500, 2009.
[24] N. A. Arafat, A. Khan, A. K. Rai, and B. Ghosh, "Neighborhood-based hypergraph core decomposition," *Proceedings of the VLDB Endowment*, vol. 16, no. 9, pp. 2061–2074, 2023.
[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[26] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. American Stat. Association*, vol. 58, p. 13–30, 1963.
[27] Y. W. Teh, D. Newman, and M. Welling, "A collapsed variational bayesian inference algorithm for latent dirichlet allocation," in *Advances in Neural Information Processing Systems*, 2007.

APPENDIX

IX. PROOF OF THEOREM 2

Proof. Consider a random variable $X_i = \hat{H}_i$ and mean = $E[\hat{H}_i]$. As entropy is bounded, $0 \leq \hat{H}_i \leq \log|Sup(P,\mathcal{G})|$. Applying Hoeffding's inequality for a bounded random variable [26]

$$Pr\left(E[\hat{H}_i] - \epsilon \leq \frac{\sum \hat{H}_i}{N} \leq E[\hat{H}_i] + \epsilon\right) \geq 1 - 2e^{\frac{-2N\epsilon^2}{\log^2|Sup(P,\mathcal{G})|}} \tag{10}$$

In order to compute $E[\hat{H}_i]$, let us assume another random variable $Y_i = \hat{Pr}_i(\mathbf{\Omega} = \Omega) = \hat{Pr}_i(\Omega)$, whose mean $\mu = E[\hat{Pr}_i(\mathbf{\Omega} = \Omega)] = Pr(\mathbf{\Omega} = \Omega)$ (Theorem 1). Similar to [27], second order Taylor approximation around $Y_i = \mu$ yields:

$$\log(Y_i) = \log(\mu) + \frac{1}{\mu}(Y_i - \mu) + \frac{1}{\mu^2}\frac{(Y_i - \mu)^2}{2} + \text{higher order terms}$$

By taking expectation and ignoring the higher order terms,

$$\begin{aligned} E[\log(Y_i)] &\approx \log(\mu) + \frac{1}{\mu}(E[Y_i] - \mu) + \frac{1}{\mu^2}\frac{E[(Y_i - \mu)^2]}{2} \\ &= \log(\mu) + \frac{1}{\mu^2}\frac{Var[Y_i]}{2} \end{aligned}$$

Replacing Y_i with $\hat{Pr}_i(\Omega)$,

$$E[\log(\hat{Pr}_i(\Omega))] = \log(Pr(\Omega)) + \frac{1}{Pr(\Omega)^2}\frac{Var[\hat{Pr}_i(\Omega)]}{2}$$

$$\implies E[-\log(\hat{Pr}_i(\Omega))] = -\log(Pr(\Omega)) - \frac{1}{Pr(\Omega)^2}\frac{Var[\hat{Pr}_i(\Omega)]}{2}$$

Multiplying both sides by $Pr(\Omega)$ and taking sum over the support set $Sup(P,\mathcal{G})$ yields,

$$\begin{aligned} &\sum_{\Omega \in Sup(P,\mathcal{G})} Pr(\Omega)E[-\log(\hat{Pr}_i(\Omega))] \\ &= \sum_{\Omega \in Sup(P,\mathcal{G})} -Pr(\Omega)\log(Pr(\Omega)) - \sum_{\Omega \in Sup(P,\mathcal{G})} \frac{1}{Pr(\Omega)}\frac{Var[\hat{Pr}_i(\Omega)]}{2} \end{aligned}$$

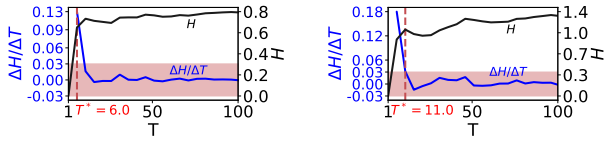
$$\implies E[E[-\log \hat{Pr}_i(\Omega)]] = H(\mathbf{\Omega}) - \sum_{\Omega \in Sup(P,\mathcal{G})} \frac{1}{Pr(\Omega)}\frac{Var[\hat{Pr}_i(\Omega)]}{2}$$

$$\implies E[\hat{H}_i] = H(\mathbf{\Omega}) - \sum_{\Omega \in Sup(P,\mathcal{G})} \frac{Var[\hat{Pr}_i(\Omega)]}{2Pr(\Omega)}$$

We compute the variance term as the following:

$$\begin{aligned} &Var[\hat{Pr}_i(\Omega)] = Var[\hat{Pr}_i(\mathbf{\Omega} = \Omega)] \\ &= \frac{1}{T^2}Var\left[\sum_{i=1}^{T} I(P(G_i) = \Omega)\right] = \frac{1}{T^2}\sum_{i=1}^{T} Var[I(P(G_i) = \Omega)] \\ &= \frac{1}{T^2}\sum_{i=1}^{T} Pr(P(G_i) = \Omega)Pr(P(G_i) \neq \Omega) \\ &= \frac{1}{T^2}\sum_{i=1}^{T} Pr(\Omega)(1 - Pr(\Omega)) = \frac{1}{T}Pr(\Omega)(1 - Pr(\Omega)) \end{aligned}$$

$$\begin{aligned} \implies &\sum_{\Omega \in Sup(P,\mathcal{G})} \frac{Var[\hat{Pr}_i(\Omega)]}{Pr(\Omega)} = \sum_{\Omega \in Sup(P,\mathcal{G})} \frac{(1 - Pr(\Omega))}{T} \\ &= \frac{|Sup(P,\mathcal{G})| - 1}{T} \\ \implies &E[\hat{H}_i] = H(\mathbf{\Omega}) - \frac{|Sup(P,\mathcal{G})| - 1}{2T} \end{aligned}$$



(a) Reach (b) SP

Fig. 5. $\Delta H/\Delta T$ vs. T (left y-axis); H vs. T (right y-axis) on *ER*

Using the above result in Equation 10,

$$Pr\left(-\frac{|Sup(P,\mathcal{G})|-1}{2T}-\epsilon \leq \frac{\sum \hat{H}_i}{N} - H(\mathbf{\Omega}) \leq -\frac{|Sup(P,\mathcal{G})|-1}{2T}+\epsilon\right)$$
$$\geq 1-2e^{\frac{-2N\epsilon^2}{\log^2|Sup(P,\mathcal{G})|}}$$
$$\implies$$
$$Pr\left(-\frac{|Sup(P,\mathcal{G})|-1}{2T}-\epsilon \leq \frac{\sum \hat{H}_i}{N} - H(\mathbf{\Omega}) \leq \frac{|Sup(P,\mathcal{G})|-1}{2T}+\epsilon\right)$$
$$\geq 1-2e^{\frac{-2N\epsilon^2}{\log^2|Sup(P,\mathcal{G})|}}$$

The last line follows since probability can only increase as we increase the upper bound. Hence, the theorem. □

Bound on the number of possible world samples. If we want to achieve a margin of error value $\frac{|Sup(P,\mathcal{G})|-1}{2T} \leq \delta \leq \frac{|Sup(P,\mathcal{G})|-1}{2T}+\epsilon$, T needs to satisfy:

$$\frac{|Sup(P,\mathcal{G})|-1}{2T} < \delta \implies T \geq \frac{|Sup(P,\mathcal{G})|}{2\delta}$$

If we aim the above with a probability $\geq \gamma$, N must satisfy:

$$1-2e^{\frac{-2N\epsilon^2}{\log^2|Sup(P,\mathcal{G})|}} \geq \gamma \implies N \geq \frac{1}{\epsilon^2}\log^2|Sup(P,\mathcal{G})|\ln\left(\frac{2}{1-\gamma}\right)$$

Corollary 1. *To achieve an error margin* $\delta \in \left(\frac{|Sup(P,\mathcal{G})|-1}{2T}, \frac{|Sup(P,\mathcal{G})|-1}{2T}+\epsilon\right)$ *with probability* γ*, it is sufficient to sample* $\frac{\ln\frac{2}{1-\gamma}}{2\delta\epsilon^2}|Sup(P,\mathcal{G})|\log^2|Sup(P,\mathcal{G})|$ *possible worlds* ($\epsilon > 0$).

X. Additional Experimental Results

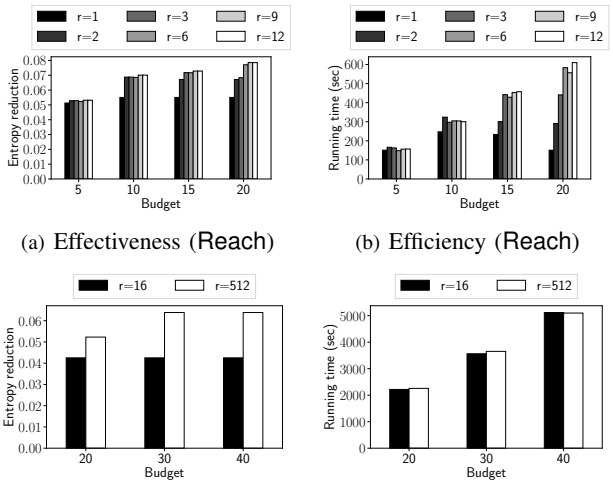
Parameters Study of the Number of Possible Worlds. We vary parameters N and T that control the number of MC-samples to assess the effect of these parameters on the estimated entropy (§IV). We demonstrate using the *ER* dataset, while reporting the optimal values of N and T for other datasets and network functions in Table IX.

Figure 5 shows the variation of entropy H and entropy-rate $\frac{\Delta H}{\Delta T}$ w.r.t. T on the *ER* dataset, while $N = 1$ remains fixed. We observe that irrespective of network functions, the entropy estimate initially fluctuates as we increase T; however, after a certain T (say $T = T^*$), it gets stable, which indicates that T^* is sufficient to approximate entropy. Note that it is difficult to determine T^* from T vs. H plot, because the range of entropy as represented in the secondary y-axis in Figure 5 is not necessarily the same for different network functions. For instance, with reachability function, $0 \leq H \leq 1$ (Figure 5(a)); however, with the shortest path distance function, it is possible that $H > 1$ (Figure 5(b)).

To address this issue, we determine T^* based on the entropy rate, represented as $\frac{\Delta H}{\Delta T} = \frac{H_{T+\delta}-H_T}{\delta}$. Here, $\delta > 0$ is the increment in T. The intuition is that after $T \geq T^*$, the rate of change in entropy $\frac{\Delta H}{\Delta T} \sim 0$. We set a threshold $\epsilon_T = 0.03$ for all network functions and datasets, and determine T^* as the point after which $|\frac{\Delta H}{\Delta T}| \leq \epsilon_T$. We select optimal N, denoted as N^*, similarly by setting a threshold $\epsilon_N = 0.001$ and computing N^* as the point after which $|\frac{\Delta H}{\Delta N}| \leq \epsilon_N$. Table IX reports N^* and T^* for all datasets with MC-sampling.

TABLE IX
Optimal N and T for all datasets with MC-sampling

graph	function	N^* ($\epsilon_N = 0.001$)	T^* ($\epsilon_T = 0.03$)
ER	SP	286	11
ER	Reach	161	6
ER	#Tri	466	11
Biomine	Reach	171	10
Biomine	SP	126	11
Flickr	#Tri	76	51
Papers	#Tri	100	60
Products	Reach	46	4



(a) Effectiveness (Reach) (b) Efficiency (Reach)

(c) Effectiveness (#Triangles) (d) Efficiency (#Triangles)

Fig. 6. Impact of budget k and hyper-parameter r on **Greedy+subgraph**, update function $\mathcal{U}_2$ (a-b) on reachability function, *products* dataset; (c-d) on #Triangles function, *papers* dataset

Parameter Study of the Greedy+subgraph Algorithm. We analyze the impact of budget k and hyper-parameter r (i.e., the number of subgraphs selected) on the performance of **Greedy+subgraph** algorithm (§V). For reachability on *products* dataset, we select 100 query pairs where each source-target pair is 5-hops away. Figure 6(a-b) demonstrates that with higher budgets, both the entropy reduction as well as the running time of the algorithm increase. For a fixed budget, as we increase r, the entropy reduction improves initially. Then, it gets saturated because the entropy has already been reduced maximally and increasing r further does not bring any benefit. In contrast, the running time usually increases with r.

For #Triangles function on *papers* dataset, we observe a similar trend in Figure 6(c-d). One important observation is that the hyper-parameter r needs to be sufficiently large to obtain significant reduction in entropy. This is because there are many triangles (e.g. 35M on *papers* dataset), and a sufficient number of triangles needs to be selected and cleaned to observe significant reduction in entropy.