

Nearest-neighbor Queries in Probabilistic Graphs

Michalis Potamias¹, Francesco Bonchi², Aristides Gionis², George Kollios¹

¹ *CS Department, Boston University
Boston, Massachusetts, USA
{mp, gkollios}@cs.bu.edu*

² *Yahoo! Research
Barcelona, Spain
{bonchi, gionis}@yahoo-inc.com*

Abstract—Large probabilistic graphs arise in various domains spanning from social networks to biological and communication networks. An important query in these graphs is the k nearest-neighbor query, which involves finding and reporting the k closest nodes to a specific node. This query assumes the existence of a measure of the “proximity” or the “distance” between any two nodes in the graph. To that end, we propose various novel distance functions that extend well known notions of classical graph theory, such as shortest paths and random walks.

We argue that many meaningful distance functions are computationally intractable to compute exactly. Thus, in order to process nearest-neighbor queries, we resort to Monte Carlo sampling and exploit novel graph-transformation ideas and pruning opportunities. In our extensive experimental analysis, we explore the trade-offs of our approximation algorithms and demonstrate that they scale well on real-world probabilistic graphs with tens of millions of edges.

I. INTRODUCTION

Uncertainty is inherent in real-world data. Driven by applications such as mobile and sensor data management, the database community has recently focused on extending the relational model to handle uncertainty. Efforts range from SQL query evaluation [1], [2], [3], [4], to ranking [5], top- k queries [6], [7], [8], [9], and mining [10], [11], [12].

In many contexts, graphs are more suited than relational tables to model and analyze the data: nodes in the graph represent entities and edges represent relationships between pairs of entities. Incorporating uncertainty leads to *probabilistic graphs*, i.e., graphs whose edges are labeled with probabilities, representing the uncertainty of their existence. Uncertainty may be a product of noisy measurements (e.g., created from sensors, or experiments) or it may represent the existence of unstable communication links. Oftentimes, the existence of an edge is predicted by some machine learning algorithm and it is inherently coupled with the confidence of the prediction. Uncertainty can also be added on purpose by means of a perturbation process aimed at ensuring privacy.

In this paper we answer k nearest-neighbor queries on probabilistic graphs. In particular, we address the following problems: what is the distance between two nodes of a probabilistic graph? Which are the k nearest neighbors of a node? What is the ranking of a set of nodes w.r.t. their distance from a specific node?

Domains of applications include the following:

Social Networks. In this context, nodes represent individuals and edges represent relations among the individuals. Uncertainty in large scale social networks may arise for many different reasons [13]. Probabilities may represent the

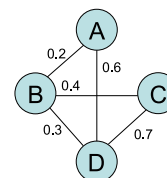


Fig. 1. A probabilistic graph.

uncertainty of a *link prediction* [14], i.e., the prediction of the future existence of edges as the network evolves. In other cases, edges are associated with probabilities representing the *influence* of one person to the other: the probability does not represent uncertainty on the edge existence itself; instead, it represents the uncertainty of the influence propagation along that edge. Indeed, in the typical setting of *viral marketing* [15], [16], the input social network is a probabilistic graph. In the context of social networks, we are interested in queries such as: “Who are the ten people who are more likely to be influenced by John?” and “Who are the five people who are more likely to be friends with Jane?”

Mobile ad hoc Networks. Consider a set of mobile nodes that move and connect to other nodes for a period of time. Since nodes are not moving randomly, but follow specific patterns, the connectivity between nodes can be estimated using measurements [17]. Therefore, we can derive the probability that a given node can deliver a packet to another node. The latter is called “delivery probability” [18]. The resulting graph is a probabilistic graph. Ranking and k nearest-neighbor queries on this probabilistic graph can be used for addressing the so called *probabilistic routing problem* [17], [18].

Biological Networks. In biology, entities such as genes, proteins, etc., are often represented as nodes in a graph and the interactions between them are modeled with edges. Since the interactions are derived by experiments that can be noisy and error-prone, each edge is associated with a confidence or uncertainty value [19]. In addition, it is possible to define a network of biological concepts using and combining different experiments and studies [20]. In this context, we want to process queries such as: “Which are the twenty proteins closest to myoglobin?”

Towards processing the aforementioned queries, we need to define novel distance functions. In order to gain more intuition on the problem, consider the probabilistic graph shown in Figure 1, which consists of four nodes and five edges. Possible

World	Probability
$\{\emptyset\}$	$(1 - p(A, B))(1 - p(A, D))(1 - p(B, C))(1 - p(B, D))(1 - p(C, D)) = 0.04032$
$\{(A, B)\}$	$p(A, B)(1 - p(A, D))(1 - p(B, C))(1 - p(B, D))(1 - p(C, D)) = 0.01008$
$\{(A, D)\}$	$p(A, D)(1 - p(A, B))(1 - p(B, C))(1 - p(B, D))(1 - p(C, D)) = 0.06048$
\vdots	\vdots
$\{(A, B), (A, D)\}$	$p(A, B)p(A, D)(1 - p(B, C))(1 - p(B, D))(1 - p(C, D)) = 0.01512$
$\{(A, B), (B, C)\}$	$p(A, B)p(B, C)(1 - p(A, D))(1 - p(B, D))(1 - p(C, D)) = 0.00672$
$\{(A, B), (B, D)\}$	$p(A, B)p(B, D)(1 - p(A, D))(1 - p(B, C))(1 - p(C, D)) = 0.00432$
\vdots	\vdots
$\{(A, B), (A, D), (B, C), (B, D), (C, D)\}$	$p(A, B)p(A, D)p(B, C)p(B, D)p(C, D) = 0.01008$

Fig. 2. Some of the possible worlds for the probabilistic graph in Figure 1.

instantiations of this graph are commonly referred to as *worlds*. In our example we have $2^5 = 32$ possible worlds given by the possible combinations of the edges. We report some of the worlds with their probability in Figure 2. The probability of a world is calculated in terms of the probabilities of existence of the various edges. For instance, assuming independence among the edge probabilities, the world in which only the two edges $\{(A, B), (B, D)\}$ exist, has a probability equal to $p(A, B)p(B, D)(1 - p(A, D))(1 - p(B, C))(1 - p(C, D))$.

Defining a meaningful scalar distance in this context is non-trivial. Suppose we are interested in quantifying the distance between B and D . A simple approach is to consider the probabilities as costs (by taking the negative logarithm of the probability) and compute the shortest-path on the resulting weighted graph. The result is the *most probable path*, which in our example is the direct path $B \rightarrow D$ and has length 1. Observe that even though in deterministic graphs we consider the length of the shortest path as the distance, in probabilistic graphs the length of the most probable path is not necessarily the most probable distance. For instance, in our example, the distance between B and D is 2 in all worlds in which (B, D) does not exist, and either $(A, B), (A, D)$ or $(B, C), (C, D)$ exist. More precisely, if we consider the distribution of the distance between B and D in terms of pairs (*distance, probability*) we have $\langle (1, 0.3), (2, 0.26), (\infty, 0.44) \rangle$. Notice that the most probable distance between B, D is infinite, and the median is 2. So what is a good definition of distance?

To answer this question, we resort to robust statistical measures that are based on the distribution of the distance over all worlds. We combine notions from statistics, such as expectations and order statistics, with notions from graph theory, such as shortest paths and random walks, and use them as building blocks. Unfortunately (but expectedly), the modelling power of probabilistic graphs makes query processing difficult.

In Section III, we show that distance functions based on shortest paths are hard to compute exactly; thus, in Section IV, we introduce efficient approximation algorithms that resort to Monte Carlo sampling. We then propose pruning algorithms to efficiently execute k nearest-neighbor queries in large probabilistic graphs (Section V). Next we define a distance function based on the notion of *Personalized PageRank* [21]. To that end, we propose an intuitive formulation of a *random walk* in probabilistic graphs and introduce a novel graph

transformation idea to efficiently approximate it (Section VI). We perform an extensive experimental study (Section VII) on real-world probabilistic graphs with tens of millions of edges: a large biological graph (BIOMINE) [20], a large coauthorship graph (DBLP), and a social networking graph (FLICKR). Our experiments explore trade-offs between accuracy and efficiency provided by our approximation algorithms. In all three datasets, queries can be processed efficiently with an accuracy loss of less than 10%.

Our main contributions are summarized as follows:

- We define meaningful distance functions on probabilistic graphs based on shortest paths and random walks.
- We propose approximation algorithms for the introduced distance functions.
- We propose pruning algorithms to efficiently address k nearest-neighbor queries on probabilistic graphs.
- We propose a novel formulation of a random walk on a probabilistic graph and show how this is equivalent to a random walk on a non-probabilistic graph.
- We investigate the trade-offs of the proposed algorithms and demonstrate their efficiency in an extensive experimental analysis with very large real-world graphs. We experimentally observe that queries are “easier” on graphs with smaller uncertainty.

II. PRELIMINARIES

Similar to normal graphs, probabilistic graphs may be undirected or directed and carry additional information on the edges such as weights. We present our ideas for the case of directed and weighted probabilistic graphs. Obviously, the same ideas apply for undirected and unweighted graphs.

Let $\mathcal{G} = (V, E, W, P)$ denote a probabilistic graph, where V denotes the set of nodes and E denotes the set of edges in the graph. The variables W and P denote respectively the weights and probabilities associated with the edge set E , and $w(e)$ and $p(e)$ denote respectively the weight and the probability of edge $e \in E$. We note that there is no need to keep information for edges with zero probability, that is, $e \in E$ if and only if $p(e) > 0$. Let G be a graph that is an instantiation of \mathcal{G} . The graph G is sampled from \mathcal{G} according to the probabilities P , that is, each edge $e \in E$ is selected to be an edge of G with probability $p(e)$. Let E_G denote the set of edges in G , then

the probability of G is:

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

We identify the probabilistic graph \mathcal{G} with the distribution $\{G\}_P$ of sampled graphs, where each of the $2^{|E|}$ possible graphs G is sampled with probability $\Pr[G]$, and we write $G \sqsubseteq \mathcal{G}$ to denote that G is sampled from \mathcal{G} (with probability $\Pr[G]$). Using the terms we used in the previous section, we can think of the probabilistic graph \mathcal{G} as a world generator process, and each graph $G \sqsubseteq \mathcal{G}$ as a *possible world*¹. Consider now two fixed nodes $s, t \in V$. Assume that given a graph G sampled from \mathcal{G} , the shortest-path distance between s and t in G is $d_G(s, t)$. We then define the distribution $\mathbf{p}_{s,t}$ of shortest-path distance between s and t , which assigns a probability value for each possible distance d as:

$$\mathbf{p}_{s,t}(d) = \sum_{G \mid d_G(s,t)=d} \Pr[G].$$

In other words, $\mathbf{p}_{s,t}(d)$ is the sum of the probabilities of all the graphs in which the shortest path distance between s and t is exactly d . Note that there may be graphs G in which s and t are disconnected. We then allow d to also take a special value ∞ , and $\mathbf{p}_{s,t}(\infty)$ is consequently defined to be the total probability of all the graphs in which s and t are disconnected.

We note that throughout the paper, it is assumed that edges are independent from one another. Some of the results that follow can be straightforwardly adjusted to handle dependencies (e.g., in Section IV, Monte Carlo sampling can be replaced with Markov Chain Monte Carlo (MCMC) sampling). However, handling general dependencies among the edges in the graph is a topic which is beyond the scope of this paper.

III. PROBABILISTIC SHORTEST PATH DISTANCE

The problem of finding shortest paths in graphs is one of the first problems that drew interest in computer science due both to its importance and to its elegant solution [22]. In this section we consider definitions of distances in probabilistic graphs that extend the standard shortest path distance. We begin our analysis by reminding the reader with a basic problem arising in probabilistic graphs.

Problem 1 (Most-Probable-Path): Given a probabilistic graph $\mathcal{G} = (V, E, W, P)$ and any pair of nodes $(s, t) \in V \times V$, find the most probable path between the nodes s and t . \square

As suggested earlier, Problem 1 can be solved easily by considering a deterministic weighted graph $G' = (V', E', W')$, with the same nodes and edges as \mathcal{G} , edge weights $w'(e) = -\log(p(e))$, and running the Dijkstra shortest-path algorithm on G' . Note that the path found may actually have very small probability, and its distance may be different than other more typical distances in sampled graphs of \mathcal{G} . For these reasons, in the rest of this section we will focus on the definition of shortest-path distance between two nodes s and t in a

¹In the remainder of the paper we use the terms *possible world*, *graph sampled from a probabilistic graph* or *possible graph* interchangeably.

probabilistic graph \mathcal{G} using the whole distribution of shortest paths $\mathbf{p}_{s,t}$, rather than any single path.

Definition 1 (Majority-Distance): Given a probabilistic graph $\mathcal{G} = (V, E, W, P)$ and any pair of nodes $(s, t) \in V \times V$ define the Majority-Distance $d_J(s, t)$ to be the most probable shortest-path distance:

$$d_J(s, t) = \arg \max_d \mathbf{p}_{s,t}(d).$$

\square

In other words, we are looking for the shortest-path distance that is the most likely to be observed when sampling a random graph from \mathcal{G} . Obviously, this problem is more interesting for unweighted graphs, or graphs with integer weights.

Based on the notion of expectation we also define:

Definition 2 (Expected-Distance): Given a probabilistic graph $\mathcal{G} = (V, E, W, P)$ and any pair of nodes $(s, t) \in V \times V$, define the Expected-Distance $d_E(s, t)$ to be the expected shortest-path distance among all possible graphs

$$d_E(s, t) = \sum_d d \cdot \mathbf{p}_{s,t}(d).$$

\square

The above definition is problematic because the expected distance is trivially infinite for interesting settings. This is due to the fact that most likely there exists a possible world where s and t are disconnected.

We modify this definition to a more meaningful one. We consider only graphs for which there exists a path between s and t . Along with the distance, we also calculate the probability of this event (*reliability*) to quantify how meaningful this expected value is. We formalize as follows:

Definition 3 (Expected-Reliable-Distance): Given a probabilistic graph $\mathcal{G} = (V, E, W, P)$ and any pair of nodes $(s, t) \in V \times V$, define the Expected-Reliable-Distance to be the expected shortest-path distance in all worlds in which there exists a path between s and t , and the probability $p(s, t)$ that there exists some path between s and t , i.e., the answer to the query (s, t) is a tuple of the form $\langle d_{ER}(s, t), p(s, t) \rangle$, where

$$d_{ER}(s, t) = \sum_{d \mid d < \infty} d \cdot \frac{\mathbf{p}_{s,t}(d)}{1 - \mathbf{p}_{s,t}(\infty)}, \text{ and } p(s, t) = \sum_{d \mid d < \infty} \mathbf{p}_{s,t}(d).$$

\square

The probability part (i.e., second item of this tuple) is known to be hard to compute exactly [23].

We also turn our attention to the median distance:

Definition 4 (Median-Distance): Given $\mathcal{G} = (V, E, W, P)$ and any pair of nodes $(s, t) \in V \times V$, define the Median-Distance $d_M(s, t)$ to be the median shortest-path distance among all possible graphs

$$d_M(s, t) = \arg \max_D \left\{ \sum_{d=0}^D \mathbf{p}_{s,t}(d) \leq \frac{1}{2} \right\}.$$

\square

Notice that the median distance may be infinite for some pairs s and t that are far away, however, it is not trivially infinite as it is in the case of the expected distance.

With respect to the hardness of computing these distance functions we provide the following remark. The problem of determining the probability that s , and t are connected, also known as the *reliability problem*, has been shown to be $\#\mathbf{P}$ -complete problem [23]. As a consequence, the problems of measuring the distances we defined above are also hard problems. Computing Expected-Reliable-Distance is a generalization of the reliability problem, so it is a $\#\mathbf{P}$ -hard problem. Consider also the problem of computing the k -th order statistic distance in the distribution $\mathbf{p}_{s,t}$, a generalization of the problem of computing Median-Distance. Solving the k -th order statistic distance problem would give a solution to the reliability problem, since one can search for the minimum k -th order statistic distance that is bounded (it is not ∞).

IV. APPROXIMATION ALGORITHMS

In this section we give approximation algorithms for computing the Expected-Reliable-Distance and Median-Distance measures in probabilistic graphs.

Expected reliable shortest path distance computation.

The exact computation of the expected reliable shortest path distance is intensive and intractable for graphs with more than some dozens of edges. It involves running a *Point-to-Point Shortest Path* algorithm to find the shortest-path distance in every graph and accumulating the expectation. At the same time, the probability of connectivity between s and t is also accumulated. This brute-force solution requires the enumeration of all graphs and thus, it is exponential to the number of edges of the graph [23].

A natural way to overcome the intractability of computing the expected reliable shortest path distance is to approximate it using sampling. In particular, we use the following sampling process:

- 1) Sample r possible graphs according to P . We will show that in theory and in practice a few hundreds of sampled graphs suffice to give us a good estimation.
- 2) Compute the shortest-path distance for the pair (s, t) in each graph. The average of these shortest-path distances is an unbiased estimator of the expected shortest-path distance. Compute also the fraction of sampled graphs in which s and t are connected, which is an unbiased estimator of the connectivity probability.

To provide a bound on the number of graphs needed to provide a good estimate for the expected reliable distance, we apply the Chernoff-bound theorem, which we reproduce below (see, e.g., [24], p.254).

Theorem 1 (Chernoff bound): Let X_1, X_2, \dots, X_r be independent and identically distributed indicator random variables, that have the same expectation $\mu = E[X_i]$. If $r \geq \frac{3}{\epsilon^2 \mu} \ln(\frac{2}{\delta})$ we have $Pr(|\frac{1}{r} \sum X_i - \mu| \geq \epsilon \mu) \leq \delta$. We say that r samples provide with an (ϵ, δ) approximation to μ . \square

We also reproduce the Hoeffding Inequality:

Theorem 2 (Hoeffding inequality): Let X_1, X_2, \dots, X_r be independent and identically distributed random variables. Assume that X_i are almost surely bounded, that is $Pr(X_i \in [a_i, b_i]) = 1$. Then for the sum of the variables $S = X_1 + \dots + X_r$ we have

$$Pr(|S - E[S]| \geq \epsilon) \leq 2 \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^r (b_i - a_i)^2}\right).$$

\square

We now state and prove the approximation of our sampling method for the Expected-Reliable-Distance problem.

Lemma 1 (Expected-Reliable-Distance): Consider a probabilistic graph $\mathcal{G} = (V, E, W, P)$ with n nodes. Consider accuracy parameters ϵ and δ , and a number of samples r . Let $\{G_i\}_P, 1 \leq i \leq r$, be a set of r graphs sampled according to P . Given a pair of vertices (s, t) , define I_i to be equal to 1 if there exists at least one path from s to t in graph G_i , and 0 otherwise. Let $G' \subseteq \{G_i\}_P$ be the set of k graphs for which $I_i = 1$, and let d_i be the corresponding shortest path distance. Then the following hold:

- 1) The random variables d_i are independent identically distributed and they are bounded in the range $[0, n - 1]$. They also have the same expectation $E[d_i] = d_{ER}(s, t)$. By selecting $r \geq \frac{(n-1)^2}{2\epsilon^2} \ln(\frac{2}{\delta})$, we have

$$Pr\left(\left|\frac{1}{r} \sum_{G_i \in G'} d_i - d_{ER}(s, t)\right| \geq \epsilon\right) \leq \delta.$$

- 2) The indicator random variables are independent identically distributed and they have the same expectation $E[I_i] = p(s, t)$. By selecting $r \geq \frac{3}{\epsilon^2 p(s, t)} \ln(\frac{2}{\delta})$ we get

$$Pr\left(\left|\frac{1}{r} \sum_{G_i} I_i - p(s, t)\right| \geq \epsilon p(s, t)\right) \leq \delta.$$

Proof: The first part of the lemma is a direct application of the Hoeffding inequality, Theorem 2. Observe that assuming connectivity, any distance in the graph takes values between $[0, n - 1]$, where n is the number of vertices in the graph. The second part of the lemma is a direct application of the Chernoff bound, Theorem 1. \blacksquare

Practical use. In order to bring Lemma 1 into practice, we introduce a threshold parameter ρ for the reliability. We make the reasonable assumption that Expected-Reliable-Distance queries with connectivity below ρ are not *interesting*. Together with accuracy parameters ϵ and δ , the parameter ρ is used to estimate the number of graphs that need to be sampled. In order to satisfy both parts of Lemma 1 we need to sample at least $r = \max\left\{\frac{3}{\epsilon^2 \rho}, \frac{(n-1)^2}{2\epsilon^2}\right\} \cdot \ln(\frac{2}{\delta})$ graphs.

The number of samples is polynomial and not exponential as the brute-force algorithm. At first glance, it can still be prohibitively large due to the factor $(n - 1)^2$. However, notice that the $(n - 1)$ comes from an estimation of the largest possible path in the network. In real-world scenarios, due to the ‘‘small-world phenomenon’’, networks have small diameter, typically smaller than 20. Thus, in reality the number of samples is much smaller. In the case of weighted graphs,

the bound needs to be adjusted by using the actual range of distances in the graph in the place of the term $(n - 1)$.

Expected median shortest path distance computation. As in the case of expected-reliable distance, computing the exact median would require enumerating all possible graphs. Since such a brute-force solution is exponential we resort again to sampling. As before, we sample a relatively small number of graphs, we compute the shortest-path distance in those graphs, and then we report the median shortest-path distance in the sample as the solution to Median-Distance. With respect to the number of samples needed to obtain an approximation to the true median distance, we have the following standard results, which is again a direct application of the Chernoff bound.

Lemma 2: Consider a total of $r \geq \frac{c}{\epsilon^2} \log(\frac{2}{\delta})$ independent samples X_1, \dots, X_r drawn from a population of N elements that has median μ , and let α, β be the elements of the population that are $\pm \epsilon N$ away from μ . Let $X = \text{median}(X_1, \dots, X_r)$. For a suitable choice of constant c we have

$$\Pr(X \in [\alpha, \beta]) > 1 - \delta.$$

□

The same sampling scheme can be used to approximate not only the median but any other quantile, for details see [25].

V. k NEAREST NEIGHBORS

In this section, we discuss the k nearest-neighbor algorithms (k -NN) for the shortest-path distance functions defined earlier. We employ pruning to drastically reduce the search space. This is combined with the Monte Carlo sampling techniques of the previous section.

A k -NN query consists of a probabilistic graph \mathcal{G} , a source node $s \in V$ and a distance function. For every other node $t \in V$, $d_P(s, t)$ is the probabilistic shortest-path distance function for which we are interested in computing the k -NN results. For instance, the distance d_P may be any of the distances d_{ER} , d_J , or d_M . For the expected reliable distance d_{ER} , we are also given a reliability threshold ρ , and we require that all returned k -NN nodes have reliability value above the threshold.

The k -NN problem is the following:

Problem 2 (k -NN): Given a probabilistic graph $\mathcal{G} = (V, E, W, P)$, a source node $s \in V$, and a probabilistic shortest-path distance d_P , find the set of k nodes $T_k(s) = \{t_1, \dots, t_k\}$ for which the distance $d_P(s, t_i)$ is less or equal to the distance $d_P(s, t)$ for any other node $t \in V \setminus T_k(s)$. □

The challenge is to compute the set $T_k(s)$ without having to compute the distance $d_P(s, t)$ for all nodes $t \in V$. We note that, depending on the application, we need to incorporate tie-break mechanisms. For example, one can break ties by (i) taking some arbitrary nodes among the ones with the same distance (ii) taking all tied-nodes and (iii) using some additional lexicographical order. Any of the previous tie-breaking mechanisms can be plugged in our algorithms.

Algorithm for the median distance. We first discuss how to obtain a k -NN algorithm for the median d_M distance function.

The algorithm is based on exploring the local neighborhood around the source node s , and computing an approximation of the distribution $\mathbf{p}_{s,t}$ truncated to the smaller distances. In particular, for a distance value D we compute the distribution $\tilde{\mathbf{p}}_{D,s,t}$, which is identical to $\mathbf{p}_{s,t}$ for all distances smaller than D and all the probability mass for distances larger than D is concentrated exactly at distance D . More precisely, we have

$$\tilde{\mathbf{p}}_{D,s,t}(d) = \begin{cases} \mathbf{p}_{s,t}(d) & \text{if } d < D \\ \sum_{x=D}^{\infty} \mathbf{p}_{s,t}(x) & \text{if } d = D \\ 0 & \text{if } d > D \end{cases}$$

Our algorithm for computing the median distance is based on the following lemma:

Lemma 3: Let $\tilde{d}_{D,M}(s, t)$ be the median distance obtained from the distribution $\tilde{\mathbf{p}}_{D,s,t}$, and $d_M(s, t)$ the actual median distance that we would have obtained from the real distribution $\mathbf{p}_{s,t}$. For any two nodes $t_1, t_2 \in V$, if it is $\tilde{d}_{D,M}(s, t_1) < \tilde{d}_{D,M}(s, t_2)$ then $d_M(s, t_1) < d_M(s, t_2)$.

Proof: First notice that $\tilde{d}_{D,M}(s, t) < D$ implies $d_M(s, t) = \tilde{d}_{D,M}(s, t)$, and $\tilde{d}_{D,M}(s, t) = D$ implies $d_M(s, t) \geq D$. Since $\tilde{d}_{D,M}(s, t_1) < \tilde{d}_{D,M}(s, t_2)$ it should be $\tilde{d}_{D,M}(s, t_1) < D$, and the lemma follows. ■

As a consequence of the above lemma, if we find the set of k nodes $T_k(s) = \{t_1, \dots, t_k, \dots\}$ for which $\tilde{d}_{D,M}(s, t_i) \leq \tilde{d}_{D,M}(s, t)$, for all $t_i \in T_k(s)$ and $t \in V \setminus T_k(s)$, we can declare the set $T_k(s)$ to be the answer to the k -NN query.

The problem is that computing the *exact* distribution $\tilde{\mathbf{p}}_{D,s,t}$ is expensive, since there are exponential many graphs to consider. We overcome this problem by sampling graphs and *approximating* the distribution $\tilde{\mathbf{p}}_{D,s,t}$. The computational saving from using the distribution $\tilde{\mathbf{p}}_{D,s,t}$ instead of $\mathbf{p}_{s,t}$ comes from the fact that for each graph sampled, the execution of the Dijkstra algorithm can be terminated as soon as nodes with distance at least D are reached.

We now describe our k -NN algorithm. The algorithm proceeds by repeating r times the following process, which is an execution of the Dijkstra algorithm up to distance D followed by an update of the distribution $\tilde{\mathbf{p}}_{D,s,t}$.

- 1) Starting from s perform a computation of the Dijkstra algorithm on \mathcal{G} . The Dijkstra algorithm *visits* the nodes in order of minimum distance discovered, and once a node is visited it never gets visited again. To apply Dijkstra in the case of probabilistic graphs, we proceed as in the case of deterministic graphs, and when it is required to explore one node we generate (sample) the out-going edges from that node. We stop the execution of the Dijkstra algorithm when we visit a node whose distance exceeds D .
- 2) For all nodes t that were visited during the execution of the Dijkstra algorithm, we know their true distances to the source node s , and this distance is less than D . Thus, we can update the distribution $\tilde{\mathbf{p}}_{D,s,t}$ by adding a counter for the distance computed in that instantiation of a sample graph. For nodes encountered but not visited, we know that their distances to s is at least D . Hence, we

can update the distribution $\tilde{\mathbf{p}}_{D,s,t}$ by updating the entry of the distribution for the lower bound distance D .

After performing the above process r times, we have an approximation of the distribution $\tilde{\mathbf{p}}_{D,s,t}$ for a subset of nodes $t \in V$. These are the nodes that were visited at least once in all of the r executions of the Dijkstra algorithm. We have no information about nodes never encountered, those are presumably nodes that are far away from s , and so we ignore them. For each node t that was visited at least once (and thus, we have kept information for the distribution $\tilde{\mathbf{p}}_{D,s,t}$) we update the entry of $\tilde{\mathbf{p}}_{D,s,t}$ that corresponds to distance D by adding the number of times that the node t was not visited. Therefore, the counts in all distributions $\tilde{\mathbf{p}}_{D,s,t}$ sum to r .

We note that the larger is the value of the parameter D , the more likely is that the condition $(\tilde{d}_{D,M}(s, t_i) \leq \tilde{d}_{D,M}(s, t))$ for $t_i \in T_k(s)$ and $t \in V \setminus T_k(s)$ will hold, and that we will obtain a solution to the k -NN problem. However, we do not know how large D needs to be. If D is very large, then the algorithm will be inefficient, since it will explore a larger neighborhood of the graph around s .

Our solution to this problem is to increase D “as you go” and to perform all the r repetitions of the Dijkstra algorithm in parallel. The algorithm proceeds in rounds, starting from distance $D = 0$, and increasing the distance by γ . In each round we resume all r executions of the Dijkstra from where they had left in the previous round. In the current round we keep visiting nodes until reaching all nodes with distance at most D . For each node t visited in any of the Dijkstra executions we update accordingly the distribution $\tilde{\mathbf{p}}_{D,s,t}$. If the distribution $\tilde{\mathbf{p}}_{D,s,t}$ of a node t reaches the 50% of its mass, then t is added to the k -NN solution. Notice that once a node is added to the solution it is never removed, because all other nodes that will be added in later steps will have greater median distances. The algorithm terminates once the solution set contains at least k nodes and the ties have been resolved.

To make the description of the Median-Distance k -NN algorithm concrete we provide the pseudocode in Algorithm 1.

Algorithm for the majority distance. The general idea of the k -NN algorithm described above, works also for the Majority-Distance. There are two main differences. First the condition of when we know for sure that a distance becomes a majority distance needs to change. In the case of median, we know that a distance will be the median once the truncated distribution $\tilde{\mathbf{p}}_{D,s,t}$ reaches the 50% of its mass. In the case of the majority, if d_1 is the current majority distance in $\tilde{\mathbf{p}}_{D,s,t}$, and r_t are all executions of Dijkstra in which a node t has been visited, the condition for ensuring that d_1 will be the majority distance for sure is $\tilde{\mathbf{p}}_{D,s,t}(d_1) \geq \frac{r-r_t}{r}$. The reason for this condition is to take care of the (worst) case that a node appears with the same distance in all executions in which it has not been encountered yet.

The second difference compared with the Median-Distance k -NN algorithm, is that it is not true anymore that a node never leaves the k -NN solution once it enters: another node t' might enter at a later step of the algorithm with a smaller majority

Algorithm 1 Median-Distance k -NN

Input: Probabilistic graph $\mathcal{G} = (V, E, W, P)$, node $s \in V$, number of samples r , number k , distance increment γ

Output: A result set T_k of k nodes for the k -NN query

- 1: $T_k \leftarrow \emptyset$
- 2: $D \leftarrow 0$
- 3: Initiate r executions of Dijkstra from s
- 4: **while** $|T_k| < k$ **do**
- 5: $D \leftarrow D + \gamma$
- 6: **for** $i \leftarrow 1 : r$ **do**
- 7: Continue visiting nodes in the i -th execution of Dijkstra until reaching distance D
- 8: For each node $t \in V$ visited update the distribution $\tilde{\mathbf{p}}_{D,s,t}$ {Create the distribution $\tilde{\mathbf{p}}_{D,s,t}$ if t has never been visited before}
- 9: **end for**
- 10: **for all** nodes $t \notin T_k$ for which $\tilde{\mathbf{p}}_{D,s,t}$ exists **do**
- 11: **if** $\text{median}(\tilde{\mathbf{p}}_{D,s,t}) < D$ **then**
- 12: $T_k \leftarrow T_k \cup \{t\}$
- 13: **end if**
- 14: **end for**
- 15: **end while**
- 16: Return T_k

distance (because the condition for deciding the majority distance for t' was satisfied at that later step). This second difference has implications on the termination condition of the Majority-Distance k -NN algorithm: the algorithm terminates not when a solution of size k is found, but once the majority distance has been decided for all nodes visited by all r executions of the Dijkstra algorithm.

VI. RANDOM WALKS IN POSSIBLE WORLDS

Random walks have been extensively studied and applied in various contexts. An excellent survey can be found in [26]. Applications of random walks range from web search [27], [28] to clustering [29] and nearest-neighbor search [30]. Based on random walks one can define many distance functions such as the hitting and commute time [26], the stationary probability of a random walk with restart [31] and Personalized PageRank [32], [33], [21].

In this section, we propose a natural definition of a random walk for general probabilistic (weighted, directed) graphs. To gain some intuition consider the following scenario: assume that a drifter finds himself in Boston and that there are three roads that he can follow, going to New York, Toronto, and Montreal respectively. Assume that each road has a proximity measure indicating the inverse of the time needed to cross them. Also, assume that each road has a probability of being open or closed, since snowfalls are not rare in the area. Now, the universe tosses coins to decide if the roads are open or closed, based on their probabilities. Assume that at that moment the roads to Toronto and Montreal are open, while the road to New York is closed. The drifter favors short roads

so he now chooses one of the two open roads to continue his journey, taking into account their proximity to Boston. If all roads are closed he must stay another night in Boston and wait for better weather the next day.

We can abstract this setting by considering a probabilistic graph $\mathcal{G} = (V, E, W, P)$, where W are edge weights denoting the *proximity* between nodes in the graph, and P denote the edge probabilities of existence. The random walk on a probabilistic graph \mathcal{G} is defined as follows: at each step of the process a new possible world (probabilistic graph instance) is generated. Some of the edges are active and some are inactive. The process starts at node u_0 and graph G_0 . At the t -th step we are at a node u_t and graph G_t . We may move to any neighbor through any active edge (u_t, v) with probability

$$\frac{w(u_t, v)}{\sum_{q|(u_t, q) \in G_t} w(u_t, q)}.$$

If there are no outgoing edges we stay at the same node. note that the probabilities P play a role only in the possible world generation, while the proximities W are the weights in the random walk step. As in the standard random walks, at each step, we either follow an edge with probability α or we teleport to a random node in the graph with probability $1 - \alpha$.

Then using the following theorem we can transform the random walk process on the probabilistic graph \mathcal{G} to a random walk process on a non-probabilistic graph $\bar{\mathcal{G}} = (V, \bar{E}, \bar{W})$. The random walk on graph $\bar{\mathcal{G}}$ is actually an equivalent definition of the process described previously, i.e., the random walk on the probabilistic graph \mathcal{G} .

Theorem 3: The probabilistic random walk on a probabilistic graph $\mathcal{G} = (V, E, W, P)$ has the same properties as a random walk on the deterministic graph $\bar{\mathcal{G}}(V, \bar{E}, \bar{W})$, where $\bar{E} = E \cup S$, with $S = \{(u, u)\}$ (i.e., the set of self-looping edges). $\bar{W} = \{\bar{w}(u, v)\}$, with

$$\begin{aligned} \bar{w}(u, u) &= \prod_{(u, q) \in E} (1 - p(u, q)), \text{ and} \\ \bar{w}(u, v) &= \sum_{G|(u, v) \in G} \frac{w(u, v)}{\sum_{(u, q) \in G} w(u, q)} \Pr[G] \end{aligned} \quad (1)$$

□

Thus, all the concepts, algorithms, and results for random walks on deterministic graphs can now be applied on $\bar{\mathcal{G}}$ (stationary distribution, PageRank, hitting times etc.).

The problem is that the complexity of computing each weight $\bar{w}(u, v)$ using the equations of Theorem 3 is exponential to the number of neighbors of each node. Thus, for graphs with nodes of high degree computing the weights $\bar{w}(u, v)$ becomes an intractable problem. Next, we provide formulas of the weights $\bar{w}(u, v)$ for various special cases, as well as approximations.

Equal weight, equal probability. Consider a graph where each edge is equally probable to appear with probability p , and all weights are equal to 1 (or to any other constant). This model is the probabilistic analogue of an Erdos-Renyi graph [34] restricted to a given topology defined by the set of edges E .

In this simple case, we can easily compute the random walk transformation. After simple algebraic calculations we get:

$$\begin{aligned} \bar{w}(u, u) &= (1 - p)^{d_u}, \text{ and} \\ \bar{w}(u, v) &= \sum_{k=1}^{d_u} \frac{\binom{d_u}{k}}{k} p^k (1 - p)^{d_u - k} = \frac{1 - \bar{w}(u, u)}{d_u}, \end{aligned}$$

where d_u denotes the out-degree of node u .

Equal weight, groups of equal probability. To build intuition, we consider the case that all edges in the graph have equal weight, while the out-going edges of a node u are partitioned into groups of equal probabilities. In particular, assume there are R groups, and let n_i be the number of edges in group i , $1 \leq i \leq R$. Also let q_i be the probability of the edges in group i . Omitting some simple algebra, the equations for the weights now become:

$$\begin{aligned} \bar{w}(u, u) &= \prod_{i=1}^R (1 - q_i)^{n_i} \\ \bar{w}(u, i) &= q_i \sum_{m_1=0}^{n_1} \dots \sum_{m_i=0}^{n_i-1} \dots \sum_{m_R=0}^{n_R} C(i, m_1, \dots, m_R) \frac{1}{1 + \sum_{j=1}^R m_j} \\ &\quad \prod_{k=1}^R q_k^{m_k} (1 - q_k)^{n_k - m_k} \end{aligned}$$

where $\bar{w}_{u,i}$ denotes the weights on all out-going edges to nodes of the group i (note that because of symmetry they have all the same weight). The function $C(i, m_1, \dots, m_R)$ gives the number of possible ways in which we can choose m_j nodes from group j , given that we have at least one node from group i . The formula is:

$$C(i, m_1, \dots, m_R) = \binom{n_1}{m_1} \dots \binom{n_i - 1}{m_i} \dots \binom{n_R}{m_R}.$$

The complexity of the algorithm implied by the equations above is $O(n_1 \cdot n_2 \cdot \dots \cdot n_R) = O((\frac{n}{R})^R)$.

In the general case, we do not have groups of edges with equal probability, so we suggest to cluster together edges with similar probabilities. In order to choose an optimal k -clustering of edges from a node u , and the respective assignment of the probability of each edge p_i to a representative probability q_k , we seek to minimize the function

$$\sum_{i=1}^{d_u} \min_{1 \leq k \leq G} (p_i - q_k)^2.$$

This problem is the 1-dimensional k -means problem and can be solved exactly in polynomial time by dynamic programming [35].

In the more general case, where edges have both probabilities and weights, we create R groups that are characterized by having similar probability and weight (q_i, t_i) . Creating such groups is casted as a 2-dimensional clustering problem, which can be solved by the k -means algorithm.

Monte Carlo sampling. We also suggest and experiment with a Monte Carlo algorithm for computing the weights $\bar{w}(u, v)$. The idea is to sample different out-going edges, for each node

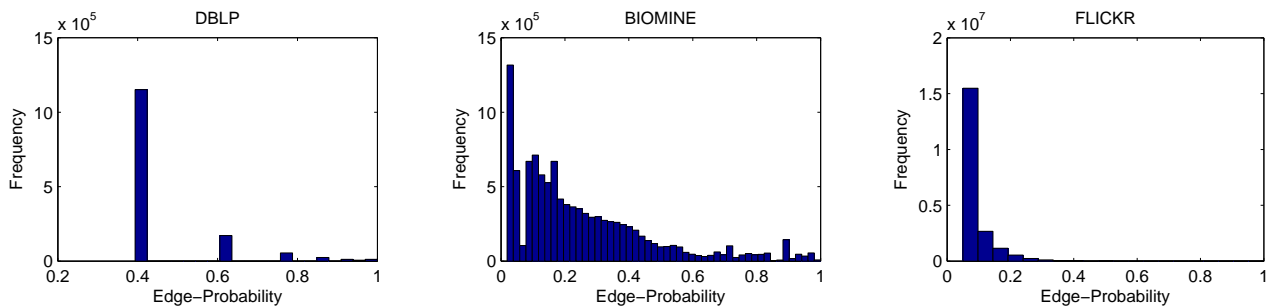


Fig. 3. Distribution of edge probabilities per dataset.

u , and estimate Equation (1) by taking the sum of probabilities over the sampled graphs only, instead of using all possible graphs. Once more, the Chernoff bound can be applied to show that a small number of samples per node u is sufficient to approximate the weights $\bar{w}(u, v)$. The number of samples depends on the probabilities and weights of the edges going out of u . We omit the details due to lack of space.

A. Random-walk k -NN algorithm

In the k -NN problem we are given a probabilistic graph \mathcal{G} and a source node s and the goal is to find the k nodes that are “nearest” to s . To that end, we consider the setting of *Personalized PageRank* (PPR) with restart to the source node s . More specifically, we perform the random walk on the probabilistic graph \mathcal{G} as defined in the previous section, but at each teleportation step, instead of restarting randomly at any node in the graph, we always restart at the source node s . The process favors visiting nodes that are close to s . The answer to the random walk k -NN problem is the set of k nodes that have the largest values of stationary distribution.

In order to compute the k -NN results for a source node s , we propose to simulate the random walk for a number of steps and approximate the stationary distribution of each node by the frequency that the node was visited during the simulation. This is a standard Monte Carlo approach for computing PageRank, see [36] for discussion and analysis of the method. In contrast with the power iteration method, the Monte Carlo approach is well-suited for the k -NN problem because it is localized in the neighborhood of the graph around s : distant nodes from s are never (or rarely) visited. Observe that we can perform the walk on $\bar{\mathcal{G}}$ instead of \mathcal{G} using Theorem 3. This way, we drastically reduce the amount of randomness needed at each step of the walk (i.e., we save the time needed to check if each outgoing edge of the currently visited node is open or closed). Of course, the trade-off is the offline computation of $\bar{\mathcal{G}}$. In addition, using the grouping technique we further speed-up the offline computation of $\bar{\mathcal{G}}$ introducing approximation error in $\bar{w}_{u,v}$. We explore experimentally these trade-offs for the random walk k -NN in Section VII. As a final remark, we note that any other technique for PPR computation on deterministic graphs (e.g., [32]) can be applied once the transformed graph $\bar{\mathcal{G}}$ has been computed.

VII. EXPERIMENTAL RESULTS

In this section we present the results of the experimental analysis of the methods in the paper. We implemented all our methods in C/ C++. All the experiments are run on a Linux server with 8 2.8GHz GHz AMD Opteron processors and 64GB of memory. We experimented on three different datasets coming from different real-world application domains.

BIOMINE: a recent snapshot of the database of the BIOMINE project [20]. The BIOMINE project is a collection of biological interactions. Interactions are directed and labelled with probabilities. We processed the original dataset to extract the connected component.

FLICKR: Flickr is a popular online community for sharing photos. Among others, users can participate in common interest groups and form friendships. We created a graph from an anonymized recent snapshot of Flickr. In particular, we extracted information about users joining interest groups. We labelled the edges with probabilities according to the following idea: assuming *homophily*, i.e., that similar interests may be an indication of users that are socially close to each other, we compute an edge probability between any two nodes (users) as Jaccard’s coefficient of the groups they belong to. This creates potentially quadratic number of edges w.r.t. the number of users. We, thus, put a threshold on Jaccard’s coefficient to be at least 0.05, and to avoid high values of the coefficient given by users who participate only in one group, we also put a threshold on the size of the intersection to be at least 3. We computed this information for a small number of users (77K) since the number of edges scales quadratically. This way we got a quite dense graph of 20M edges.

DBLP: we extracted the DBLP coauthors graph from a recent snapshot of the DBLP database of journal publications. There is an undirected edge between two authors if they have coauthored a journal paper. We labelled the edges with probabilities using an exponential distribution. The number of coauthored papers between every two authors was used as the original information.

The datasets follow a power-law out-degree distribution. We present the degree distribution of BIOMINE in Figure 4 as an example and note that the others are similar. Observe that there are some central nodes, connected to 5% of the database. All the probabilistic graphs, in their whole, are connected, but obviously, many disconnected worlds exist and thus infinite

TABLE I
DATASETS CHARACTERISTICS.

Dataset	$ V $	$ E $	$MaxOutdegree$
DBLP	226K	1.4M	238
BIOMINE	1M	10M	139603
FLICKR	77K	20M	5765

TABLE II
FREQUENCY OF INFINITY DISTANCES.

	BIOMINE	DBLP	FLICKR
MAJORITY	0.83	0.78	0.42
EXPECTED-RELIABLE	0.69	0.56	0.35
MEDIAN	0.69	0.56	0.35

distances. Table I summarizes size and maximum out-degree of the datasets. We also plot histograms with the frequencies that various probability values occur on edges in Figure 3. Notice that DBLP has a few probability values due to the generating process. Observe also how Flickr probability values are generally very small again due to the generation process, while BIOMINE has a more balanced probability distribution.

A. Shortest Path based Distances

Recall from Section IV that in order to compute the median, the majority, and the expected reliable distance in practice we perform sampling of worlds. In order to assess the quality of the sampling we performed the following experiment. We accumulated distances running the full BFS traversal for 500 sampled nodes on a sample of 500 worlds. We present the distributions of all the distance functions in Figure 5. For the expected reliable distance we set the reliability threshold to 0.5. We removed the infinity bars from the histograms and refer to them in Table II for completeness.

Note that all distances have similar distributions and that they look qualitatively similar to distributions of shortest path distances in non-probabilistic scale free networks. Table II indicates that there are many infinite distances in these datasets. Recall from Figure 4 that there are many nodes with one or two edges. Also recall from Figure 3 that these edges most likely have low probability. In other words, these nodes are disconnected from the main part of the graph in most worlds generated by the probabilistic graph. Thus their median, expected-reliable and majority distances to the rest are oftentimes infinite.

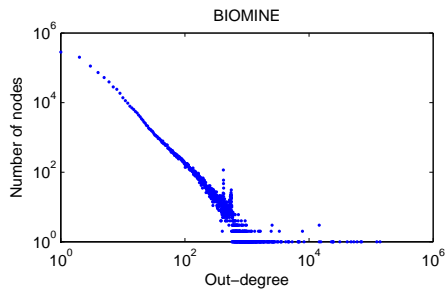


Fig. 4. Out-degree distribution of BIOMINE.

We move on to study the convergence of the distance functions based on the number of worlds. In Figure 6, we plot the Mean Squared Error of the distance approximations (using the distances according to a sample of 500 worlds as the “ground truth”), for various numbers of worlds. Observe that they all converge as expected to 0. We conclude that 200 worlds were enough to compute distances accurately since the Mean Squared Error drops below 0.2 for all datasets and all distances. Note that for sake of presentation we have removed from consideration all distances that were infinite.

B. k NN Pruning

In this part, we present results from our experimental evaluation of the pruning algorithms introduced in Section V. We implement the algorithms for both the median and the majority distances. Tie-breaking is done by extending $T_k(s)$ to include all objects tied with t_k . We experiment with the two most important components of the algorithm: efficiency and quality of the results. We measure efficiency for each run of a k -NN algorithm as a fraction of the number of nodes visited over all executions of the Dijkstra algorithm, over the total number of nodes in the graph. The reason is that the visited nodes are the ones for which we keep histogram information and, thus, the ones involved in the computation of the k -NN result. Other aspects of efficiency such as number of worlds sampled can be taken into account and factored in the presented graphs.

Figure 7(a) illustrates the fraction of nodes visited as a function of k for the Median-Distance k -NN problem for a sample of 200 worlds. The efficiency decreases sublinearly as k increases. Figure 7(b) plots the fraction of nodes visited as a function of the number of worlds sampled for the Majority-Distance k -NN problem for the 10NN problem. As expected, efficiency decreases with the number of worlds but it stabilizes for some hundreds of worlds. In both plots, the three datasets behave in a similar way.

In Figure 7(d), we present the stability of the k -NN result set for the median distance. Our experiment is as follows: we first compute the result of 50NN for the Median-Distance problem for 1000 worlds and we consider this to be the “ground truth”. Then we compute k -NN results as a function of number of worlds and we compute Jaccard’s coefficient in each case with the “ground truth”. We observe again that the solution stabilizes for a few hundreds of worlds.

In order to study the effect of the edge-probability values on the pruning efficiency we conduct the following experiment. We boost each edge’s probability p , by making it $p_d = 1 - (1 - p)^d$. In other words, we give each edge d chances to be instantiated, instead of 1. For $d = 1$, we have $p_1 = p$, while for $d > 1$, we have $p_d > p$. We plot the pruning efficiency in Figure 7(c) with respect to parameter d for the 50NN median experiment and 200 worlds. Observe that the pruning efficiency depends heavily on the uncertainty of the edges. In particular, decreasing the uncertainty results to dramatic increase in the pruning power for all datasets. Observe in Figure 3 that FLICKR and BIOMINE bear more

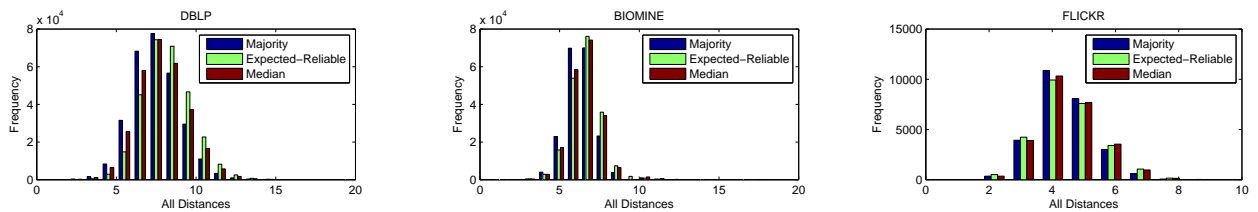


Fig. 5. Distribution of majority, expected-reliable and median distances

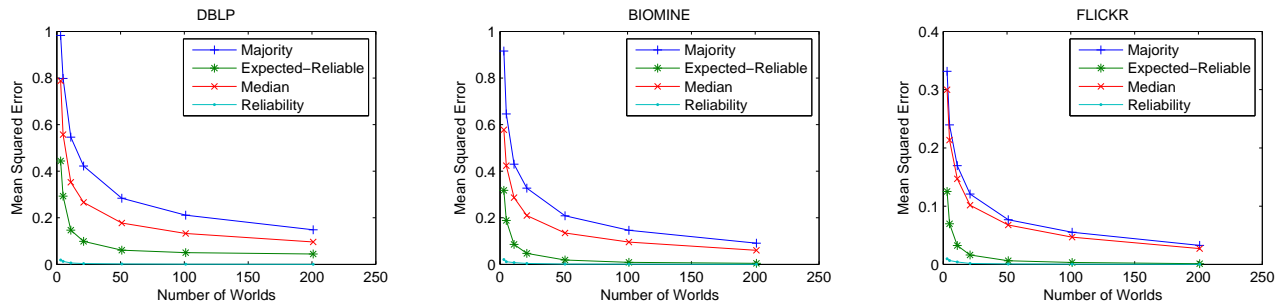


Fig. 6. Convergence of the distance approximations.

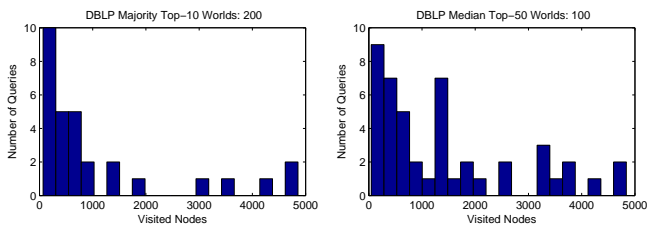


Fig. 8. k -NN pruning: distribution of the number of nodes visited.

uncertainty. This explains DBLP’s superior performance in all pruning experiments. We conclude that the smaller the edge-probabilities the harder the pruning task.

Finally, we take a closer look to the visited nodes during pruning. The figures in all previous plots are computed as the average over 100 queries. Figure 8 illustrates the distribution of the nodes visited during k -NN computation for the DBLP dataset with respect to the number of queries. In most cases the number of nodes visited is small. However, when the k -NN result is not found quickly, a large number of nodes end up being visited.

C. Probabilistic Random Walk

In Section VI, we showed the equivalence of the probabilistic random walk to a random walk on a deterministic weighted graph. The exact computation of the transformed graph is performed locally at each node, using only its outgoing edges. However, it scales exponentially to the number of those edges, making it hard to compute exactly for out-degrees that are greater than 30. We implemented the exact computation using log probabilities for numerical stability. Also, we implemented the grouping algorithm, which reduces the complexity by introducing error, but it is still impractical for more than 10 groups and 100 edges. Finally, we implemented the sampling algorithm (with an option to group edges).

Amount of MC sampling. In Figure 9, we present the performance of the Monte Carlo sampling in terms of success for the k -NN query. Success in the k -NN query is computed using Jaccard’s coefficient for the k -NN sets of our method and the true k -NN. Note that we can only estimate the true k -NN using many samples, since computing the actual probabilities of the edges is exponential to the number of edges. We used k equal to 10, 20, and 50. Regarding efficiency the transformation scales linearly to the number of samples and it can take as low as a few minutes (for 1000 samples) to a few hours (for 50000 samples) using one CPU. However, we remark that the transformation can straightforwardly be parallelized since it is local to a node and its edges. In order to compute the stationary distribution of the Personalized PageRank we performed 1M random walks per experiment after empirically observing that this number was large enough. The teleport probability has been set to 0.20. Note that since the ground truth cannot be found these graphs have been plotted using as ground truth the result of taking 50K samples. Notice that 50K samples yield more than 90% accuracy in BIOMINE and DBLP. In FLICKR which is a more “volatile” graph since it is very dense and has extremely low probability edges, the performance is around 80% at 50K samples, and one needs to sample 200K worlds to reach 90% performance.

Number of Groups. We perform an experiment to gain intuition about the error introduced when we force edges to participate in groups of equal probability. We present our results for various numbers of groups in Figure 9. As expected DBLP converges very fast (4 groups are enough). Recall from Table I that the maximum out-degree is just 238; on the other hand BIOMINE and FLICKR which have nodes with out-degree in the thousands need more groups to converge. Still, we get the surprising result that 20 groups are enough. Thus, the offline computation of the transformation can be safely sped up for nodes with large outdegree, using the

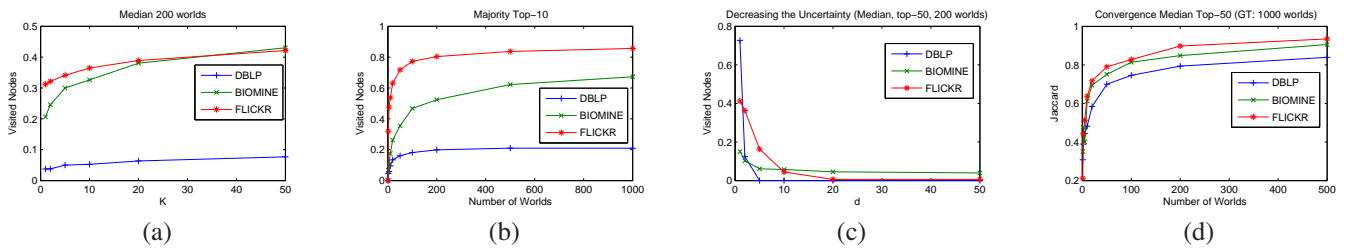


Fig. 7. Number of visited nodes with respect to k (a) and number of worlds (b). Plot (c) illustrates the relationship between pruning and edge-probabilities. Plot (d) shows the stability of the method.

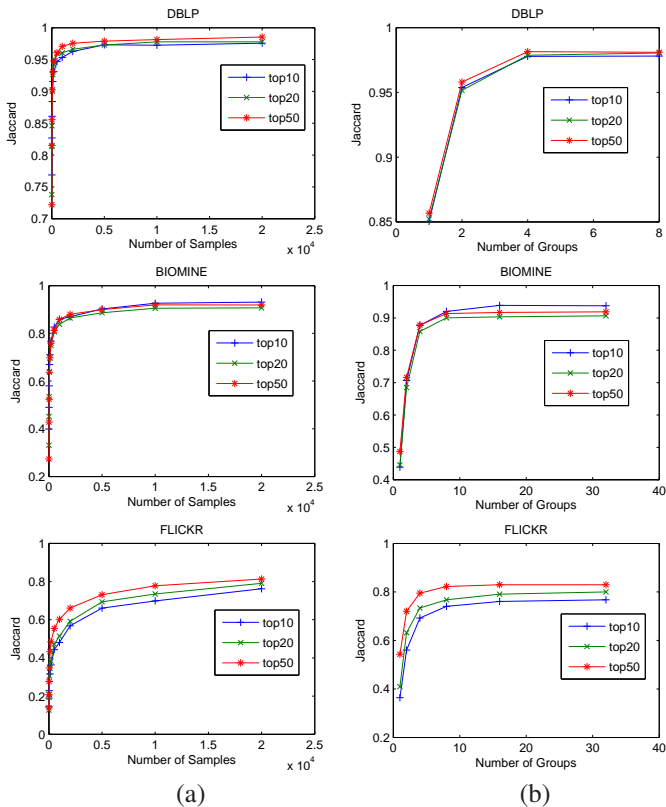


Fig. 9. (a) Performance in k -NN vs Monte Carlo parameter. (b) Performance in k -NN vs Number of Groups.

grouping technique. We note that for this experiment we used everywhere 20K MC samples.

VIII. RELATED WORK

Probabilistic databases have received increased interest recently and a number of system prototypes have been developed already that can store, manage, and query probabilistic data. Notable examples include the MayBMS [2], MystiQ [1], ORION [3] and Trio [4]. All of these systems extend relational operations and approaches to deal with uncertainty using possible world semantics. Since computing exact answers to many typical SQL queries has been shown to have $\#\mathbf{P}$ -complete data complexity [1], most of the recent works have concentrated on computing approximate answers [7], [37].

Another important area in probabilistic databases is the definition and efficient evaluation of top- k queries (similar to our k -NN queries). Soliman et al. were the first to define

meaningful top- k queries in probabilistic databases [6]. Since then, a number of different definitions of top- k queries have been proposed, as well as methods to evaluate them efficiently [5], [8], [38], [39], [40], [41], [42], [43]. A unified approach that can express and generalize many of the proposed top- k definitions and deal with correlated tuples has also appeared recently [44]. The work of Re et al. on computing the top- k tuples on a specific set of SQL queries [7] is closely related to this paper. The authors use an extension of the Monte Carlo sampling algorithm. The central design is to use multiple instances of the Monte Carlo algorithm running in parallel with the goal being to find a good approximation of the tuple probabilities that will be used to eventually find the top- k .

Our work on probabilistic shortest paths is related to the Stochastic Shortest Path problem (SSP) that has been studied in the field of Operations Research. This line of research deals with computing the probability density function (aka pdf) of the shortest path length for a pair of nodes [45]. By contrast, we avoid the exact computation of the pdf of a source node to all other nodes (which in our datasets are millions) since it is not a scalable solution for the k -NN problem under investigation. Our pruning algorithms for the median and majority shortest path problems are tailored to compute as little of the pdf as possible for the smaller possible fraction of nodes with no loss in accuracy. In [46], the problem of finding a shortest path on a probabilistic graph is addressed by transforming each edge weight to its expected value and running Dijkstra. Clearly in our setting this expectation is always infinite. Others investigate the pdf computation over various cost functions [47], [48], [49]; these are interesting directions for extending our work. Another interesting approach is to revisit the probabilistic graph model; e.g., Jaillet has considered a model with node failures [50].

Finally, in a different context, graph databases have also received a lot of attention due to many applications in social network and scientific applications [51], [52]. However, these works assume deterministic graphs and they do not deal with probabilistic graphs and their distances.

IX. CONCLUSION

Probabilistic graphs are a natural representation in many application domains, ranging from mobile ad hoc networks to social and biological networks. In this paper, we address the problem of processing k nearest-neighbor queries in large probabilistic graphs. To that end, we study distance notions

based on shortest-paths and random-walks. We provide a set of meaningful functions and we show that they are hard to compute. Thus, we introduce approximation algorithms that resort to Monte Carlo sampling and novel graph-transformation ideas. We move on to introduce pruning algorithms for the kNN problem. We apply our algorithms on three real-world probabilistic graphs and observe experimentally that smaller uncertainty leads to more effective pruning during query processing. Overall, our extensive empirical analysis confirms, on the one hand, the meaningfulness of our measures, and on the other hand, the efficiency and accuracy of our approximation methods. Future work involves generalizing the current model and the algorithms to handle correlations [53], node failures [50], and arbitrary probability distributions.

Acknowledgment. We are grateful to Hannu Toivonen for the BIOMINE dataset.

REFERENCES

- [1] N. N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," in *VLDB*, 2004.
- [2] L. Antova, T. Jansen, C. Koch, and D. Olteanu, "Fast and simple relational processing of uncertain data," in *ICDE*, 2008.
- [3] S. Singh, C. Mayfield, S. Mittal, S. Prabhakar, S. E. Hambrusch, and R. Shah, "Orion 2.0: native support for uncertain data," in *SIGMOD*, 2008.
- [4] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom, "Trio: A system for data, uncertainty, and lineage," in *VLDB*, 2006.
- [5] G. Cormode, F. Li, and K. Yi, "Semantics of ranking queries for probabilistic data and expected ranks," in *ICDE*, 2009.
- [6] M. Soliman, I. Ilyas, and K. C.-C. Chang, "Top-k query processing in uncertain databases," in *ICDE*, 2007.
- [7] C. Re, N. N. Dalvi, and D. Suciu, "Efficient top-k query evaluation on probabilistic data," in *ICDE*, 2007.
- [8] K. Yi, F. Li, G. Kollios, and D. Srivastava, "Efficient processing of top-k queries in uncertain databases with x-relations," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 12, pp. 1669–1682, 2008.
- [9] M. Hua, J. Pei, W. Zhang, and X. Lin, "Ranking queries on uncertain data: a probabilistic threshold approach," in *SIGMOD*, 2008.
- [10] G. Cormode and A. McGregor, "Approximation algorithms for clustering uncertain data," in *PODS*, 2008.
- [11] C. Aggarwal, Y. Li, J. Wang, and J. Wang, "Frequent pattern mining with uncertain data," in *KDD*, 2009.
- [12] M. Renz, T. Bernecker, F. Verhein, A. Zuefle, and H.-P. Kriegel, "Probabilistic frequent itemset mining in uncertain databases," in *KDD*, 2009.
- [13] E. Adar and C. Re, "Managing uncertainty in social networks," *IEEE Data Eng. Bull.*, vol. 30, no. 2, pp. 15–22, 2007.
- [14] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *CIKM*, 2003.
- [15] P. Domingos and M. Richardson, "Mining the network value of customers," in *KDD*, 2001.
- [16] D. Kempe, J. M. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003.
- [17] S. Biswas and R. Morris, "Exor: opportunistic multi-hop routing for wireless networks," in *SIGCOMM*, 2005.
- [18] J. Ghosh, H. Ngo, S. Yoon, and C. Qiao, "On a routing problem within probabilistic graphs and its application to intermittently connected networks," in *INFOCOM*, 2007.
- [19] S. Asthana, O. D. King, F. D. Gibbons, and F. P. Roth, "Predicting protein complex membership using probabilistic network reliability," *Genome Research*, vol. 14, pp. 1170–1175, 2004.
- [20] P. Sevón, L. Eronen, P. Hintsanen, K. Kulovesi, and H. Toivonen, "Link discovery in graphs derived from biological databases," in *DILS*, 2006.
- [21] G. Jeh and J. Widom, "Scaling personalized web search," in *WWW*, 2003.
- [22] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, December 1959.
- [23] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal on Computing*, vol. 8, no. 3, pp. 410–421, 1979.
- [24] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, January 2005.
- [25] G. S. Manku, S. Rajagopalan, and B. G. Lindsay, "Approximate medians and other quantiles in one pass and with limited memory," in *SIGMOD*, 1998.
- [26] L. Lovász, "Random walks on graphs: A survey," in *Combinatorics, Paul Erdős is Eighty*, vol. 2, 1993, pp. 1–46.
- [27] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford Digital Library Technologies Project, Tech. Rep., 1998.
- [28] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.
- [29] H. Qiu and E. R. Hancock, "Clustering and embedding using commute times," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1873–1890, 2007.
- [30] P. Sarkar, A. W. Moore, and A. Prakash, "Fast incremental proximity search in large graphs," in *ICML*, 2008.
- [31] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *ICDM*, 2006.
- [32] D. Fogaras and B. Rácz, "Towards scaling fully personalized pagerank," *Algorithms and Models for the Web-Graph*, pp. 105–117, 2004.
- [33] D. Fogaras and B. Rácz, "Practical algorithms and lower bounds for similarity search in massive graphs," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 5, pp. 585–598, 2007.
- [34] P. Erdős and A. Rényi, "On random graphs, i," *Publicationes Mathematicae (Debrecen)*, vol. 6, pp. 290–297, 1959.
- [35] R. Bellman, "On the approximation of curves by line segments using dynamic programming," *Communications of ACM*, vol. 4, no. 6, 1961.
- [36] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova, "Monte carlo methods in pagerank computation: When one iteration is sufficient," *SIAM Journal of Numerical Analysis*, Tech. Rep., 2005.
- [37] C. Koch, "Approximating predicates and expressive queries on probabilistic databases," in *PODS*, 2008.
- [38] X. Lian and L. Chen, "Probabilistic ranked queries in uncertain databases," in *EDBT*, 2008.
- [39] C. Jin, K. Yi, L. Chen, J. X. Yu, and X. Lin, "Sliding-window top-k queries on uncertain streams," *PVLDB*, vol. 1, no. 1, pp. 301–312, 2008.
- [40] R. Cheng, L. Chen, J. Chen, and X. Xie, "Evaluating probability threshold k-nearest-neighbor queries over uncertain data," in *EDBT*, 2009.
- [41] M. A. Soliman and I. F. Ilyas, "Ranking with uncertain scores," in *ICDE*, 2009.
- [42] X. Zhang and J. Chomicki, "On the semantics and evaluation of top-k queries in probabilistic databases," in *ICDE Workshops (DBRank)*, 2008.
- [43] T. Ge, S. Zdonik, and S. Madden, "Top-k queries on uncertain data: On score distribution and typical answers," in *To Appear in SIGMOD*, 2009.
- [44] J. Li, B. Saha, and A. Deshpande, "A unified approach to ranking in probabilistic databases," in *To Appear in VLDB*, 2009.
- [45] H. Frank, "Shortest path in probabilistic graphs," *Operations Research*, vol. 17, no. 4, pp. 583–599, July-August 1969.
- [46] G. Dantzig, *Linear Programming and Extensions*. Princeton University Press, August 1998.
- [47] L. Deng and M. D. F. Wong, "An exact algorithm for the statistical shortest path problem," in *ASP-DAC '06: Proceedings of the 2006 conference on Asia South Pacific Design Automation*, 2006.
- [48] D. Sarioz and V. Dan, "The expected shortest path problem: algorithms and experiments," *J. Comput. Small Coll.*, vol. 16, no. 4, pp. 311–312, 2001.
- [49] D. Rasteiro and J. Anjo, "Optimal paths in probabilistic networks," *Journal of Mathematical Sciences*, vol. 120, no. 2, pp. 974–987, 2004.
- [50] P. Jaillet, "Shortest path problems with nodes failures," *Networks*, vol. 22, pp. 589–605, 1992.
- [51] Y. Tian, J. M. Patel, V. Nair, S. Martini, and M. Kretzler, "Periscope/gq: a graph querying toolkit," *PVLDB*, vol. 1, no. 2, pp. 1404–1407, 2008.
- [52] P. Boldi and S. Vigna, "The webgraph framework ii: Codes for the world-wide web," in *Data Compression Conference*, 2004.
- [53] P. Sen and A. Deshpande, "Representing and querying correlated tuples in probabilistic databases," in *ICDE*, 2007.