# k-Nearest Neighbors in Uncertain Graphs

Michalis Potamias[1] Francesco Bonchi[2] Aristides Gionis[2] George Kollios[1]

[1]Computer Science Department
Boston University, USA
{mp,gkollios}@cs.bu.edu

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

## ABSTRACT

Complex networks, such as biological, social, and communication networks, often entail uncertainty, and thus, can be modeled as *probabilistic graphs*. Similar to the problem of similarity search in standard graphs, a fundamental problem for probabilistic graphs is to efficiently answer $k$-*nearest neighbor* queries ($k$-NN), which is the problem of computing the $k$ closest nodes to some specific node.

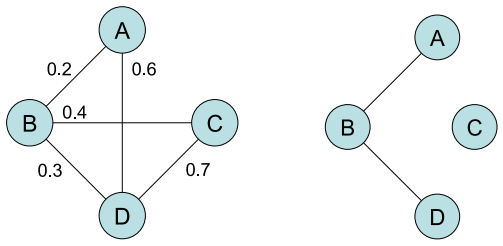
In this paper we introduce a framework for processing $k$-NN queries in probabilistic graphs. We propose novel distance functions that extend well-known graph concepts, such as shortest paths. In order to compute them in probabilistic graphs, we design algorithms based on sampling. During $k$-NN query processing we efficiently prune the search space using novel techniques.

Our experiments indicate that our distance functions outperform previously used alternatives in identifying true neighbors in real-world biological data. We also demonstrate that our algorithms scale for graphs with tens of millions of edges.

## 1. INTRODUCTION

Noisy measurements, inference models, and privacy preserving perturbation processes produce uncertain data. Research in probabilistic relational databases has focused on SQL query evaluation [3, 4, 13, 39], mining [2, 12, 35], ranking and top-$k$ queries [11, 20, 34, 40, 45, 46]. However, in many prevalent application domains, such as social, biological, and mobile networks, graphs serve as better models than relational tables. Incorporating uncertainty to graphs leads to probabilistic graphs.

Biological networks constitute one of the main applications of probabilistic graphs. Nodes represent genes and proteins, and edges represent *interactions* among them. Since the interactions are derived through noisy and error-prone lab experiments, each edge is associated with an uncertainty value [5]. In protein-protein interaction networks [26], possible interactions have been established experimentally for a limited number of pairs of proteins. Identifying protein neighbors in such interaction networks is useful for predicting possible co-complex memberships [26, 30] and new interactions [38]. Thus, $k$-nearest neighbor queries can be used to provide candidate links, whose validity can be further explored through strenuous biological experiments.




(a) (b)

**Figure 1: (a) A probabilistic graph with 5 edges and $2^5 = 32$ possible worlds. (b) One possible world: $G = \{(A,B),(B,D)\}$. Probability of $G$ is $Pr[G] = p(A,B)\,p(B,D)\,(1-p(A,D))\,(1-p(B,C))\,(1-p(C,D)) = 0.2 \times 0.3 \times 0.4 \times 0.6 \times 0.3 = 0.00432$.**

In large social networks uncertainty arises for various reasons [1]. The probability of an edge may represent the uncertainty of a *link prediction* [28] or the influence of a person to another, as for example in *viral marketing* [15, 24]. Thus, in the context of social-network applications, we are interested in queries such as: "Who are the ten people that Alice influences the most?"

In mobile ad-hoc networks, mobile nodes move and connect to each other. The connectivity between nodes can be estimated using measurements [8], and the notion of the "delivery probability" can be used to quantify the probability that a given node can deliver a packet to another node [19]. Therefore, $k$-nearest neighbor queries can be used for addressing the *probabilistic-routing problem* [8, 19].

The problems of computing distance functions and processing $k$-NN queries are fundamental for probabilistic graphs, just as they are for standard graphs. They serve as primitive operators for tasks such as link prediction, clustering, classification, and graph mining. In this paper, we present a principled extension of these problems in the presence of uncertainty and we assess the quality of the proposed distance functions using a real probabilistic protein-protein interaction network.

To gain more intuition on the graph model and the difficulties in defining a meaningful distance function, we present


*VLDB '10,* September 13-17, 2010, Singapore

a simple example. Consider the probabilistic graph shown in Figure 1(a). Each edge is associated with a probability of being present. Possible instantiations of this graph are commonly referred to as *worlds* [13]. In our example, there are $2^5 = 32$ possible worlds. The probability of a world is calculated based on the probability of its edges as shown in the example in Figure 1(b).

Suppose we are interested in quantifying the distance between two nodes, say, $B$ and $D$. The Most-Probable-Path-Distance has already been used as a distance function [38]; it is defined as the length of the most probable path. For the pair $(B, D)$, it is the direct path $B \rightarrow D$ and its length is 1. Another trivial alternative is to consider the probability that there exists a path from $B$ to $D$ (i.e., *reliability*) as an indicator of closeness. In our example this is the probability that at least one of the three possible paths between $B$ and $D$ exist, i.e., $1 - (1 - 0.3)(1 - 0.12)(1 - 0.28) = 0.56$. Now, consider the distribution of the shortest path distance between $B$ and $D$, in terms of pairs $(distance, probability)$: i.e., $\langle (1, 0.3), (2, 0.26), (\infty, 0.44) \rangle$. Notice that the *most probable* distance is infinite, while the *median* distance is 2. The *expected* distance, disregarding the infinity part, is 1.46.

Among the five different measures presented which ones are the most appropriate? How are they computed? We address these questions in the remainder of this paper.

Our contributions in this paper are summarized as follows:

1. We extend shortest paths and random walks to define meaningful distance functions in probabilistic graphs.
2. We show that our distance functions outperform their competitors (i.e, MostProbPath and Reliability) in identifying true neighbors, in real-world data.
3. We define the $k$-NN problem in probabilistic graphs and introduce novel pruning algorithms to address it.
4. We propose a novel probabilistic random walk and show its equivalence to a standard random walk.
5. We perform an extensive experimental evaluation with large real-world graphs. We show that our algorithms work in practice, and we observe that smaller probability values result in greater processing cost.

## 2. PROBABILISTIC GRAPH MODEL

In this section, we formally present the data model considered in this paper. We assume independence among edges noting that most of our results are applicable to graphs with edge correlations (see Appendix B for details). Similar to normal graphs, probabilistic graphs may be undirected or directed and carry additional labels on the edges (such as weights). For sake of generality, we focus on directed and weighted probabilistic graphs. We assume that the weights are discrete.

Let $\mathcal{G} = (V, E, P, W)$ be a probabilistic graph, where $V$ and $E$ denote the set of nodes and edges respectively. The variable $P$ denotes the probabilities associated with the edges; $p(e)$ denotes the probability of edge $e \in E$. In the case of *weighted* graphs, we will use $W$ to collectively denote the weights, and $w(e)$ for the weight of an edge. Let $G$ be a graph that is 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 $p(e)$. If $E_G$ denotes the set of edges of $G$, then the probability associated with $G$ 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 $\Pr[G]$). 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*.

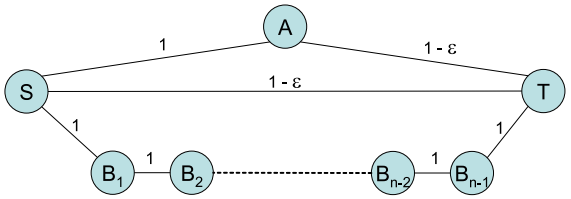



**Figure 2: The most probable path may be long.**

## 3. PROBABILISTIC GRAPH DISTANCE

Next we extend the concept of shortest paths and random walks to distance functions in probabilistic graphs.

First, we remind the reader that the MostProbPath distance can be computed 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 $G'$ [38].

The definition of MostProbPath distance has several limitations. First the probability of such a path may be arbitrarily small. Second, even if the probability of the path itself is large, the probability that it is indeed the shortest path can be arbitrarily small. Consider the example of Figure 2. The lower path between $S$ and $T$ has length $n$, which can be arbitrarily large, and probability 1.

However, the probability that it will be a shortest path is close to 0, i.e., $2\epsilon - \epsilon^2$, since most likely, one of the paths of length 1 or 2 will be present. In this paper, we overcome the limitations of the MostProbPath distance by using statistics of the *shortest path distance distribution*, rather than any single path. Furthermore, in Section 5, we demonstrate experimentally that our distance functions outperform the MostProbPath distance in real world link prediction scenarios.

### 3.1 Shortest path distribution

Given $G \sqsubseteq \mathcal{G}$, let the shortest-path distance between $s$ and $t$ be $d_G(s, t)$. We define the distribution $\mathbf{p}_{s,t}$ of shortest-path distance between $s$ and $t$ 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 worlds in which the shortest path distance between $s$ and $t$ is exactly $d$. Note that there may be worlds $G$ in which $s$ and $t$ are disconnected. Thus, we allow $d$ to take a special value $\infty$, and $\mathbf{p}_{s,t}(\infty)$ is consequently defined to be the total probability of all the worlds in which $s$ and $t$ are disconnected. We base our distance function definitions on standard statistical measures of the distribution $\mathbf{p}_{s,t}$.

DEFINITION 1 (MEDIAN-DISTANCE). *Given a probabilistic graph $\mathcal{G} = (V, E, P, W)$ and any two nodes $s$ and $t$, we define the Median-Distance $d_{\mathrm{M}}(s,t)$ to be the median shortest-path distance among all possible worlds*

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

Note that the median distance may be infinite for some pairs $s$ and $t$ and that all results presented in this paper for Median-Distance, hold for any $k$-th order statistic as well.

Another commonly adopted statistic is the majority distance, which is the shortest-path distance that is most likely to be observed when sampling a random graph from $\mathcal{G}$:

DEFINITION 2 (MAJORITY-DISTANCE). *Given a probabilistic graph $\mathcal{G} = (V, E, P, W)$ and any two nodes $s$ and $t$ we define the Majority-Distance $d_{\mathrm{J}}(s,t)$ to be the most probable shortest-path distance:*

$$d_{\mathrm{J}}(s,t) = \arg\max_{d} \mathbf{p}_{s,t}(d).$$

In the case of *weighted* graphs, Majority-Distance is meaningful if the weights come from a discrete domain.

Next we consider the notion of expectation. In most cases, the expected shortest path distance is trivially infinite due to the presence of $\infty$. Thus, we seek a more meaningful definition. We consider only the events that a graph contains a path between $s$ and $t$. Taking expectation over the non-infinite distances gives the following:

DEFINITION 3 (EXPECTED-RELIABLE-DISTANCE). *Given a probabilistic graph $\mathcal{G} = (V, E, P, W)$ and any two nodes $s$ and $t$ we define the Expected-Reliable-Distance to be the expected shortest-path distance in all worlds in which there exists a path between $s$ and $t$. We also define the probability $p(s,t)$ that there exists some path between $s$ and $t$.*

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

Computing the Expected-Reliable-Distance is a #**P**-hard problem since it is a generalization of the reliability problem [43].

## 3.2 Probabilistic random walk

We define a distance function based on random walks. In contrast with shortest-path functions which rely on one path, random walks consider all paths. Also, their navigational choices are random instead of optimal. Random walks have already been used for nearest-neighbor search in standard graphs [36].

Our distance function is based on the *Individual PageRank* (IPR) [16]. Given a node $s$, IPR refers to the stationary distribution of a PageRank walk [31] that always teleports back to $s$, instead of teleporting to any node in the graph.

We now define the IPR for probabilistic graphs. We consider a *weighted* probabilistic graph $\mathcal{G} = (V, E, W, P)$, where $W$ denotes the *proximity* between nodes in the graph, and $P$ denotes the edge probabilities. The walk is parameterized on the source node $s$ and a teleportation probability $a$.

The walk is initialized at node $s$ and world $G_0$, which is sampled online according to $P$. At the $t$-th step, the walk is characterized by the current node $u_t$ and the current world $G_t$. At the $t$-th step we either follow an active edge (with probability $1-a$) or we teleport back to $s$ (with probability $a$). In the first scenario, we follow an 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. We call this process Probabilistic-Random-Walk. We define Random-Walk-Distance as (see Appendix C for examples and details):

DEFINITION 4 (RANDOM-WALK-DISTANCE). *Given $\mathcal{G} = (V, E, W, P)$, a teleportation probability $\alpha$, and any pair of nodes $(s,t) \in V \times V$ define the Random-Walk-Distance $d_{\mathrm{RW}}(s,t)$ to be the inverse of the stationary probability of $t$ of the Probabilistic-Random-Walk with source $s$.*

# 4. $K$-NEAREST NEIGHBORS

In the following we introduce algorithms to efficiently process $k$-NN queries for the distance functions defined previously. First, we show how sampling can be used to compute the distances defined in the previous section.

## 4.1 Computing the distance functions

The exact computation of the Median-Distance is intractable, as it involves executing a point-to-point shortest-path algorithm in every world and taking the median. A natural way to overcome the intractability of computing the Median-Distance is to approximate it using sampling. The idea is to $(i)$ sample $r$ possible graphs according to $P$, and $(ii)$ compute the median of the shortest-part distances in the sample graphs. Using the Chernoff bound we have the following standard result:

LEMMA 1. *Consider $r \geq \frac{c}{\epsilon^2}\log(\frac{2}{\delta})$ independent samples $X_1, \ldots, X_r$ drawn from a population of $N$ elements, which has median $\mu$. Let $\alpha$ and $\beta$ be the elements of the population that are $\pm\epsilon N$ away from $\mu$. Let $X = \mathrm{median}(X_1, \ldots, X_r)$. For a suitable choice of the constant $c$ we have*

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

The Expected-Reliable-Distance distance can also be efficiently approximated via sampling (see Appendix A for additional details). Also, we remark that Lemma 1 works on graphs with conditional probabilities on their edges and we refer the reader to Appendix B for additional details.

## 4.2 $k$-NN problem definition

A $k$-NN query on a probabilistic graph $\mathcal{G}$, consists of a source node $s$, a probabilistic distance function $d_{\mathrm{P}}(s,t)$, and $k$. The distance $d_{\mathrm{P}}$ may be any of the distances $d_{\mathrm{J}}$, $d_{\mathrm{M}}$, $d_{\mathrm{ER}}$, or $d_{\mathrm{RW}}$. The $k$-NN problem is the following:

PROBLEM 1 ($k$-NN). *Given $\mathcal{G} = (V, E, P, W)$, a source node $s$, a probabilistic distance $d_{\mathrm{P}}$, and $k$, find the set of $k$ nodes $T_k(s) = \{t_1, \ldots, t_k\}$ for which the distance $d_{\mathrm{P}}(s,t_i)$ is less or equal to the distance $d_{\mathrm{P}}(s,t)$ 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 $d_{\mathrm{P}}(s,t)$ for all nodes $t \in V$. We note that various tie-break mechanisms may be incorporated in this definition, depending on the application.

Next, we present pruning algorithms for Median-Distance and Majority-Distance.

## 4.3 Median-distance $k$-NN pruning

The algorithm for $d_{\mathrm{M}}$ is based on exploring the local neighborhood around the source node $s$, and computing the distribution $\mathbf{p}_{s,t}$, *truncated* to the smaller distances. In particular,

for a distance value $D$, we compute the distribution $\mathbf{p}_{D,s,t}$, which is identical to $\mathbf{p}_{s,t}$ for all distances smaller than $D$. The remaining probability mass is concentrated exactly at distance $D$. More precisely, we have:

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

Our algorithm is based on the following lemma:

LEMMA 2. *Let $d_{D,\mathrm{M}}(s,t)$ be the median distance obtained from the distribution $\mathbf{p}_{D,s,t}$, and $d_{\mathrm{M}}(s,t)$ be the actual median distance that we would have obtained from the real distribution $\mathbf{p}_{s,t}$. For any two nodes $t_1, t_2 \in V$, $d_{D,\mathrm{M}}(s,t_1) < d_{D,\mathrm{M}}(s,t_2)$ implies $d_{\mathrm{M}}(s,t_1) < d_{\mathrm{M}}(s,t_2)$.*

PROOF. First notice that $d_{D,\mathrm{M}}(s,t) < D$ implies $d_{\mathrm{M}}(s,t) = d_{D,\mathrm{M}}(s,t)$, and $d_{D,\mathrm{M}}(s,t) = D$ implies $d_{\mathrm{M}}(s,t) \geq D$. Since $d_{D,\mathrm{M}}(s,t_1) < d_{D,\mathrm{M}}(s,t_2)$ it should be $d_{D,\mathrm{M}}(s,t_1) < D$, and the lemma follows. □

A direct corollary of the above lemma is that if we find the set of $k$ nodes $T_k(s) = \{t_1, \ldots, t_k, ...\}$ for which $d_{D,\mathrm{M}}(s,t_i) \leq d_{D,\mathrm{M}}(s,t)$, 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. This is the core idea of our pruning scheme.

Computing the exact distribution $\mathbf{p}_{D,s,t}$ is expensive, since there are exponentially many graphs to consider. We overcome this problem by sampling graphs and approximating the distribution $\mathbf{p}_{D,s,t}$, with the sample distribution $\tilde{\mathbf{p}}_{D,s,t}$.[1] Observe that Lemma 2 holds for any fixed sample of worlds, by replacing all distributions and distance values with their sample based approximations.

The algorithm (whose pseudocode is provided in Algorithm 1) proceeds by repeating the following process $r$ times:

1. Starting from $s$, we perform a computation of the Dijkstra algorithm. Once a node is visited it never gets visited again. To apply Dijkstra in probabilistic graphs, we proceed as in the case of deterministic graphs: when it is required to explore one node we generate (sample) the outgoing edges from that node. We stop the execution of the algorithm when we visit a node whose distance exceeds $D$.
2. For all nodes $t$ that were visited we either update or instantiate their distribution $\tilde{\mathbf{p}}_{D,s,t}$. Their distance is less than $D$.

After performing the above process $r$ times, we have computed 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. We have no information about nodes never encountered, those are presumably nodes that are far away (in terms of $d_{\mathrm{M}}$) from $s$ and we can safely ignore them. Note that after the $r$ traversals are done, for each node $t$ that was visited at least once, the entry of $\tilde{\mathbf{p}}_{D,s,t}$ that corresponds to distance $D$ is set to the number of traversals that the node $t$ was *not* encountered. Therefore, the counts in all distributions $\tilde{\mathbf{p}}_{D,s,t}$ sum to $r$.

We note that the larger the value of the parameter $D$, the more likely that the condition $(\tilde{d}_{D,\mathrm{M}}(s,t_i) \leq \tilde{d}_{D,\mathrm{M}}(s,t)$ for $t_i \in T_k(s)$ and $t \in V \setminus T_k(s))$ holds, and a solution to the $k$-NN problem is obtained. However, we do not know exactly how large $D$ needs to be. Our solution to this problem is to increase $D$ as you go and to perform all $r$ repetitions of the Dijkstra algorithm in parallel. The algorithm proceeds in rounds, starting from distance $D = 0$, and increasing the distance by $\gamma$. In each round, we resume all $r$ executions of the Dijkstra from where they had left in the previous round, and pause them when they reach all nodes with distance at most $D$. 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. All other nodes that will be added in later steps will have greater or equal median distances. The algorithm terminates once the solution set contains at least $k$ nodes. This scheme works for any order statistic other than the median.

**Algorithm 1** Median-Distance $k$-NN

**Input:** Probabilistic graph $\mathcal{G} = (V, E, P, W)$, node $s \in V$, number of samples $r$, number $k$, distance increment $\gamma$
**Ouput:** $T_k$, a result set of $k$ nodes for the $k$-NN query

```
1: T_k ← ∅; D ← 0
2: Initiate r executions of Dijkstra from s
3: while |T_k| < k do
4:    D ← D + γ
5:    for i ← 1 : r do
6:       Continue visiting nodes in the i-th execution
         of Dijkstra until reaching distance D
7:       For each node t ∈ V visited
         update the distribution p̃_{D,s,t} {Create the distribution
         p̃_{D,s,t} if t has never been visited before}
8:    end for
9:    for all nodes t ∉ T_k for which p̃_{D,s,t} exists do
10:      if median(p̃_{D,s,t}) < D then
11:         T_k ← T_k ∪ {t}
12:      end if
13:   end for
14: end while
```

## 4.4 Majority-distance $k$-NN pruning

The $k$-NN algorithm for Majority-Distance is similar to the one for Median-Distance. There are two main differences: In the case of the median, the distance of a node $t$ from $s$ is determined once the truncated distribution $\tilde{\mathbf{p}}_{D,s,t}$ reaches the 50% of its mass. In the case of the majority, let $d_1$ be the current majority value in $\tilde{\mathbf{p}}_{D,s,t}$, and let $r_t$ be all Dijkstra executions in which a node $t$ has been visited. The condition for ensuring that $d_1$ will be the exact majority distance is $\tilde{\mathbf{p}}_{D,s,t}(d_1) \geq \frac{r-r_t}{r}$. The above conditions take care of the (worst) case that a node will appear with the same distance value in all future Dijkstra executions.

The second difference is in the termination condition; a node that enters the $k$-NN set, may not be in the final result: another node might enter at a later step of the algorithm with a smaller majority distance. Candidate nodes can be discarded if their majority distance is guaranteed to be greater than the largest distance in the $k$-NN set.

# 5. QUALITATIVE ANALYSIS

In the previous sections we defined distance functions among nodes that reside in probabilistic graphs and proposed algorithms to compute distance and $k$-NN queries. Before we

[1] For the remainder of the paper, we denote approximations using the symbol ˜.

explore the efficiency of these algorithms (Section 6), we showcase that our distance functions outperform their alternatives RELIABILITY and MOSTPROBPATH in identifying *true* neighbors via a link-prediction experiment. We experiment with two real-world datasets, a protein-protein interaction network and a co-authorship graph. Both datasets consist of two parts: a *probabilistic graph* and a set of *ground-truth* neighbors. The underlying assumption is that a *good* distance function measured on the graph should be better correlated with the ground-truth neighbors.

We perform the following *classification* experiment. We choose a random ground-truth edge $(A, B_0)$, and a random node $B_1$ from the graph, such that $(A, B_1)$ is *not* a ground-truth edge. We then randomly permute $B_0$ and $B_1$, producing either a triplet of class 0 of the form $\langle A, B_0, B_1\rangle$ or a triplet of class 1, of the form $\langle A, B_1, B_0\rangle$. Given a triplet, the classification task is to assign the appropriate class. In other words, the classifier attempts to identify the true neighbor. We build various classifiers based on different distance functions. All classifiers are unaware of the ground truth and pick the node that appears to be closer to $A$ in the probabilistic graph.

For both datasets we used 8000 random triplets. We used 50 random worlds for MEDIAN, MAJORITY, EXPECTEDREL, and RELIABILITY. We used a reliability threshold equal to 0 for EXPECTEDREL, without optimization. Next, we present the specific experimental setups and the results.

## 5.1 Protein-protein interaction network

We used the protein-protein interaction network (PPI) created by Krogan et al. [26]. Two proteins are linked if it is likely that they interact. The network consists of 3 672 proteins and 14 317 interactions labeled with probabilities. As ground truth we used the complex-membership lists from the MIPS database [30]. Keeping only the part of MIPS with nodes that are also in the PPI network, we had a set of 13 024 *ground-truth* co-complex memberships with 867 distinct proteins.

Finding good neighbors in a PPI network is important. The PPI network consists of uncertain interactions that have been established using biological experiments. The $k$-NN sets can be used as a filter step to identify candidate neighbors that will be validated experimentally. Our experiment demonstrates that our functions produced higher quality neighbors than their competitors, and thus, they constitute better choices for $k$-NN in PPI networks.

## 5.2 Co-authorship network

We predict co-authorships in the DBLP after the year 2001, based on historical data. We took a recent snapshot of the DBLP database. Two authors are linked if they have coauthored a journal or a conference paper. In order to obtain a probabilistic graph, we isolated the co-authorships that occurred before 2001. Each edge was labeled with the number of papers coauthored before 2001. We generated probabilities based on these values using the intuition that the greater the weight of the edge, the more likely to be re-activated in the future. Regarding the value of the edge probabilities, we note that the first coauthored paper between two authors is more informative than any subsequent. So, we applied an exponential cdf of mean 2 to the weights. This way, weight of 1 is mapped to probability of 0.39, 2 to 0.63, and 10 to 0.99. We emphasize that we did not perform

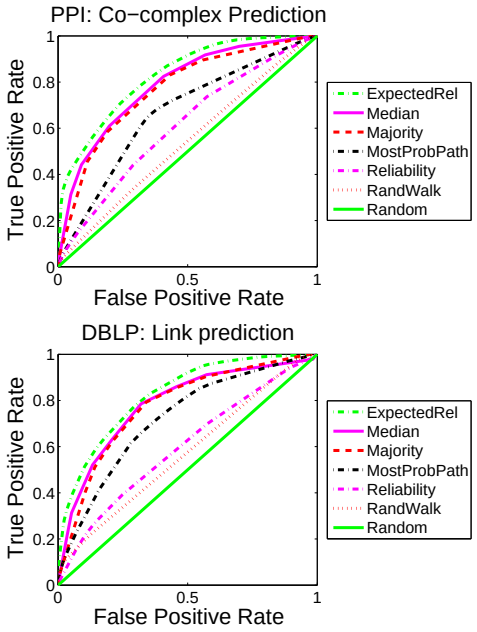



**Figure 3: ROC curves. ExpectedRel, Median, and Majority dominate the rest.**

any optimization on estimating the probabilities; finding the optimal probability model is beyond the scope of this paper. The resulting probabilistic graph consists of 90 713 nodes and 462 242 edges. The ground truth is formed by taking all edges formed after 2001. After discarding authors that have not appear before 2001, and edges that appear both before and after 2001, we obtained 90 946 ground-truth edges with 23 966 distinct authors.

## 5.3 Results

We illustrate our results using ROC curves in Figure 3. In both PPI and DBLP datasets, our functions EXPECTEDREL, MEDIAN, MAJORITY clearly dominate the rest. In contrast with their main competitor, MOSTPROBPATH, which essentially treats probabilities as weights and relies on the length of just one path of the probabilistic graph, our functions take into account the possible-world semantics and use the entire shortest-path length distribution. Therefore, they yield higher quality results. For instance, in the PPI network, for a *false positive rate* (FPR) of 0.2, all three yield a *true positive rate* (TPR) above 0.6, while the MOSTPROBPATH yields a TPR below 0.4. The RANDWALK and RELIABILITY functions are clearly worse than the ones based on shortest paths, but still more informative than the baseline RANDOM classifier. Contrary to all of our distance functions, both competitors do not work for weighted probabilistic graphs, since they do not take weights into account. For example, the most probable path could have an arbitrarily large weight in a *weighted* probabilistic graph. To be fair, we use *unweighted* probabilistic graphs for both experiments. We remark that even though EXPECTEDREL dominates marginally MEDIAN and MAJORITY, the appropriate function for other datasets could be different. Thus, it should be chosen based on a similar qualitative analysis as the one presented above.

We conclude that our distance functions are better in identifying true neighbors in real world data than their competitors. Also, this experiment demonstrates that our dis-

tance functions differ from one another. Finally, the remarkable difference between our functions and their competitors draws future directions for research in applications, where MostProbPath is currently used [38].

## 6. EFFICIENCY ANALYSIS

We next report empirical assessment of the efficiency of the methods presented in this paper. We implemented all our methods in C++. All the experiments were run on a Linux server with 8 2.8GHz GHz AMD Opteron processors and 64GB of memory.

We tested the performance of our algorithms on three datasets from different real-world application domains: BIOMINE, FLICKR, and DBLP.

**BIOMINE.** This is a recent snapshot of the database of the BIOMINE project [38], which is a collection of biological interactions. Interactions are directed and labeled with probabilities.

**FLICKR.** Flickr is a popular online community for sharing photos. Among other activities, 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 labeled the edges with probabilities assuming *homophily*, the principle that similar interests may indicate social ties. Namely, we computed the edge probability between any two nodes (users) as the Jaccard coefficient of the groups they belonged to. This process creates quadratic number of edges with respect to the number of users. We, thus, put a threshold of 0.05 to the probability value. In order 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), and we obtained a dense graph of 20M edges.

**DBLP.** We created a DBLP graph by considering an undirected edge between two authors if they have coauthored a journal paper. We labeled the edges with probabilities as described in Section 5.

Figure 4(a) shows the edge-probability distributions in the three datasets. Notice that DBLP has only a few probability values. Observe also that Flickr probability values are generally very small, while BIOMINE has a more uniform probability distribution. Additional details about the datasets can be found in the Appendix D.

We accumulated distances running the full BFS traversal for 500 sampled nodes on a sample of 500 worlds. We set the sample-size to 500 worlds after experimentally observing that the result was stable. We present the distributions of all the distance functions in Figure 4(b). For the expected reliable distance we set the reliability threshold to 0.5 (we have removed the infinity bars from the histograms (see Appendix E for details). Observe that all distance functions yield similar distributions. Also, they all look qualitatively similar to typical distributions of shortest path distances in non-probabilistic networks with *scale-free* characteristics.

We move on to study the convergence of the distance functions based on the number of worlds. In Figure 5 we plot the Mean Squared Error (MSE) 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

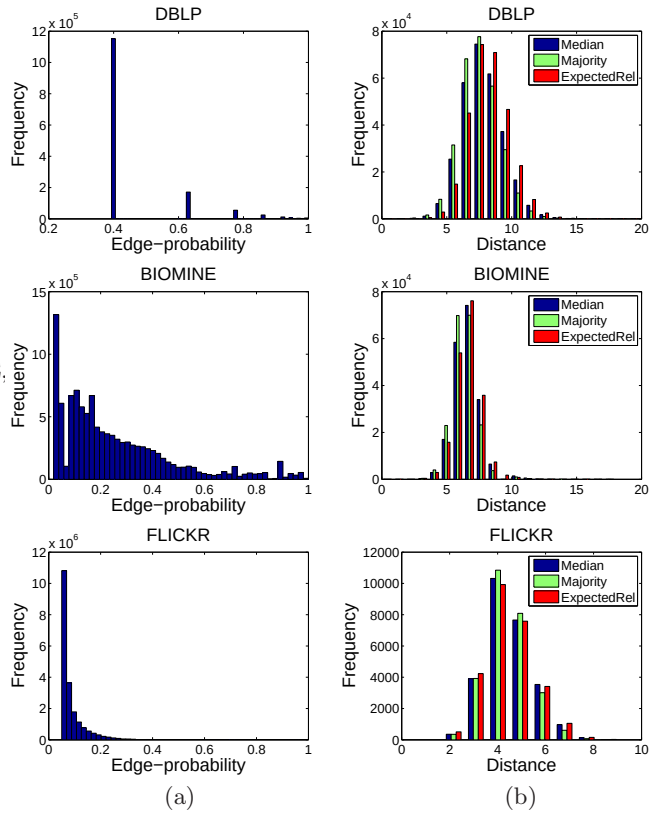



**Figure 4: Distribution of (a) edge probabilities, (b) distances.**

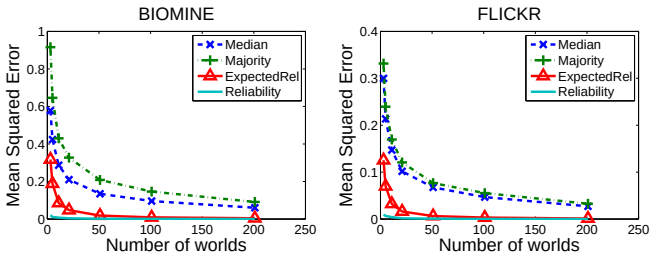



**Figure 5: MSE vs. worlds. 200 worlds are enough.**

that 200 worlds are enough to compute distances accurately since the MSE drops below 0.2 for all datasets and all distances. Even though we had already established in theory that a small number of samples is needed, it was surprising to find out that 200 worlds are enough, in datasets with tens of millions of edges.

### 6.1 $k$-NN pruning

We present an experimental evaluation of the pruning algorithms introduced in Section 4.2. We implemented the algorithms for both the median and the majority distances. Tie-breaking was 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 the union of the visited nodes in all executions of the Dijkstra algorithm,

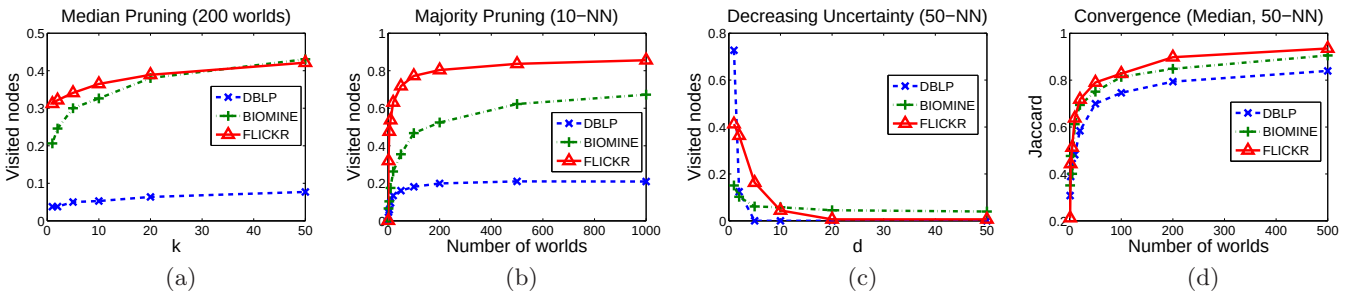



**Figure 6: Number of visited nodes with respect to (a) $k$ and (b) number of worlds. (c) Pruning efficiency vs. edge-probabilities. (d) Convergence of the method.**

**Table 1: Pruning Speedup.**

| MEDIAN, *200 Worlds* | | | | |
|---|---|---|---|---|
| $k$ | 5 | 10 | 20 | 50 |
| DBLP | 269 | 267 | 208 | 185 |
| BIOMINE | 247 | 183 | 121 | 95 |
| FLICKR | 111 | 102 | 81 | 66 |
| MAJORITY, *10-NN* | | | | |
| Worlds | 20 | 50 | 100 | 200 |
| DBLP | 18 | 22 | 22 | 23 |
| BIOMINE | 55 | 59 | 59 | 65 |
| FLICKR | 3.6 | 3.6 | 3.8 | 4.0 |

over the total number of nodes in the graph. The reason is that the number of visited nodes determines the cost (node retrieval and histogram maintenance). Other aspects of efficiency, such as the number of worlds sampled, can be taken into account and factored in the presented graphs.

Figure 6(a) shows the fraction of visited nodes as a function of $k$ for the median $k$-NN problem and 200 worlds. The efficiency decreases sublinearly as $k$ increases. Note that a node is counted as *visited* if it is visited in at least one of the worlds. Figure 6(b) shows the fraction of visited nodes as a function of the number of worlds sampled for the majority 10-NN problem. As expected, efficiency decreases with the number of worlds but it stabilizes for some hundreds of worlds. In both plots, all three datasets yield similar behavior. We also measured wall-clock times in CPU ticks and report the speedup of our pruning techniques in Table 1 (averaged over 100 queries). Observe that the gains are large, and that they decrease as $k$ increases. For example, computing the median 5-NN with pruning and with 200 worlds in BIOMINE was 247 times faster than without pruning; it took 0.5 seconds instead of 123. The wall-clock gains with respect to the number of worlds were almost constant.

In Figure 6(d) we present the stability of the $k$-NN result for the median distance, 50-NN. We considered the result in 1000 worlds as the ground truth since it was stable. Clearly, the solution stabilizes for a few hundred worlds.

Finally, to study the effect of the edge-probability values on the pruning efficiency, we conducted the following experiment: we boosted each edge's probability $p$, by making it $p_d = 1-(1-p)^d$. Thus, we gave each edge $d$ chances to be instantiated, instead of one. For $d = 1$, we have $p_1 = p$, while for $d > 1$, we have $p_d > p$. We plot the pruning efficiency in Figure 6(c) with respect to parameter $d$ for the 50-NN median experiment and 200 worlds. Clearly, the pruning efficiency depends heavily on the uncertainty of the edges; increasing the probabilities results to dramatic increase in the pruning power for all datasets. We conclude that the smaller the edge-probabilities the harder the pruning task. Observe also in Figure 4(a) that FLICKR bears more uncertainty (lower probability values). This explains the superior performance of DBLP and BIOMINE in the runtime experiments in Table 1.

## 7. RELATED WORK

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 [17]. 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 [14], the problem of finding a shortest path on a probabilistic graph is addressed by transforming each edge's pdf to its expected value and running Dijkstra. Clearly in our setting this expectation is always infinite. Others investigate the pdf computation over various application-dependent cost functions [33], while Jaillet has considered a model with node failures [21].

Recently, probabilistic databases have received increased interest and a number of system prototypes have been developed that can store and query probabilistic data. Notable examples include the BayesStore [44], MayBMS [4], MCDB [22], MystiQ [13], ORION [39], PrDB [37] and Trio [3]. These systems model data with relations and therefore, they cannot perform shortest path computations on graphs efficiently. Also, since computing exact answers to many typical SQL queries has been shown to have #**P**-complete data complexity [13], research has focused on computing approximate answers [25, 34].

Another important area in probabilistic relational 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 [40]. Since then, a number of different definitions of top-$k$ queries have been proposed, as well as methods to evaluate them efficiently [10, 11, 18, 23, 41, 45, 47]. A unified approach that can express and generalize many of the proposed top-$k$ definitions has appeared recently [27].

Probabilistic-Random-Walk extends random walks, which

have been extensively studied [29]. Applications of random walks range from web search [31] to clustering [32] and nearest-neighbor search [36].

Finally, the need to store and query massive graph data has lead to an increased interest in graph databases [9, 42]. The focus here is on standard graphs, not on probabilistic.

# 8. CONCLUSION

Probabilistic graphs are a natural representation in many applications, ranging from mobile ad-hoc networks to social and biological networks. In this paper, we addressed the problem of processing nearest-neighbor queries in large probabilistic graphs. To that end, we extended the concepts of shortest paths and random walks in standard graphs. We defined meaningful distance functions and introduced approximation algorithms based on sampling. Our algorithms prune the search space by computing a truncated version of the shortest-path distance distribution.

We assessed the quality of our functions in real-world data. Our functions identify better neighbors than their competitors. In addition, our extensive empirical analysis confirmed the efficiency of our methods. We also observed that larger probabilities of the edges result to more effective $k$-NN pruning. Future work involves enriching our framework with more powerful models that can handle node failures [21] and arbitrary probability distributions.

**Acknowledgments.** George Kollios and Michalis Potamias were partially supported by the NSF IIS-0812309 grant. We are grateful to Hannu Toivonen for the BIOMINE dataset and to Konstantin Voevodski for his valuable help.

# 9. REFERENCES

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

[2] C. Aggarwal, Y. Li, J. Wang, and J. Wang. Frequent pattern mining with uncertain data. In *KDD*, 2009.

[3] 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.

[4] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *ICDE*, 2008.

[5] S. Asthana, O. D. King, F. D. Gibbons, and F. P. Roth. Predicting protein complex membership using probabilistic network reliability. *Genome Research*, 14:1170–1175, 2004.

[6] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova. Monte carlo methods in pagerank computation: When one iteration is sufficient. Technical report, SIAM Journal of Numerical Analysis, 2005.

[7] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] S. Biswas and R. Morris. Exor: opportunistic multi-hop routing for wireless networks. In *SIGCOMM*, 2005.

[9] P. Boldi and S. Vigna. The webgraph framework ii: Codes for the world-wide web. In *Data Compression Conference*, 2004.

[10] R. Cheng, L. Chen, J. Chen, and X. Xie. Evaluating probability threshold k-nearest-neighbor queries over uncertain data. In *EDBT*, 2009.

[11] G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, 2009.

[12] G. Cormode and A. McGregor. Approximation algorithms for clustering uncertain data. In *PODS*, 2008.

[13] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, 2004.

[14] G. Dantzig. *Linear Programming and Extensions*. Princeton University Press, August 1998.

[15] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, 2001.

[16] D. Fogaras and B. Rácz. Towards scaling fully personalized pagerank. *Algorithms and Models for the Web-Graph*, pages 105–117, 2004.

[17] H. Frank. Shortest path in probabilistic graphs. *Operations Research*, 17(4):583–599, July-August 1969.

[18] T. Ge, S. Zdonik, and S. Madden. Top-k queries on uncertain data: On score distribution and typical answers. In *SIGMOD*, 2009.

[19] 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.

[20] M. Hua, J. Pei, W. Zhang, and X. Lin. Ranking queries on uncertain data: a probabilistic threshold approach. In *SIGMOD*, 2008.

[21] P. Jaillet. Shortest path problems with nodes failures. *Networks*, 22:589–605, 1992.

[22] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. M. Jermaine, and P. J. Haas. Mcdb: a monte carlo approach to managing uncertain data. In *SIGMOD Conference*, pages 687–700, 2008.

[23] C. Jin, K. Yi, L. Chen, J. X. Yu, and X. Lin. Sliding-window top-k queries on uncertain streams. *PVLDB*, 1(1), 2008.

[24] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

[25] C. Koch. Approximating predicates and expressive queries on probabilistic databases. In *PODS*, 2008.

[26] N. J. Krogan, G. Cagney, and al. Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, 440(7084):637–643, March 2006.

[27] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1):502–513, 2009.

[28] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM*, 2003.

[29] L. Lovász. Random walks on graphs: A survey. In *Combinatorics, Paul Erdős is Eighty*, 1993.

[30] H. Mewes and al. Mips: analysis and annotation of proteins from whole genomes. *Nucleic Acids Res*, 32:D41–44, 2004.

[31] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[32] H. Qiu and E. R. Hancock. Clustering and embedding using commute times. *IEEE PAME*, 29(11), 2007.

[33] D. Rasteiro and J. Anjo. Optimal paths in probabilistic networks. *Journal of Mathematical Sciences*, 120(2), 2004.

[34] C. Re, N. N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, 2007.

[35] M. Renz, T. Bernecker, F. Verhein, A. Zuefle, and H.-P. Kriegel. Probabilistic frequent itemset mining in uncertain databases. In *KDD*, 2009.

[36] P. Sarkar, A. W. Moore, and A. Prakash. Fast incremental proximity search in large graphs. In *ICML*, 2008.

[37] P. Sen, A. Deshpande, and L. Getoor. PrDB: managing and exploiting rich correlations in probabilistic databases. *VLDB Journal*, 2009.

[38] P. Sevon, L. Eronen, P. Hintsanen, K. Kulovesi, and H. Toivonen. Link discovery in graphs derived from biological databases. In *DILS*, 2006.

[39] 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.

[40] M. Soliman, I. Ilyas, and K. C.-C. Chang. Top-k query processing in uncertain databases. In *ICDE*, 2007.

[41] M. A. Soliman and I. F. Ilyas. Ranking with uncertain scores. In *ICDE*, 2009.

[42] Y. Tian, J. M. Patel, V. Nair, S. Martini, and M. Kretzler. Periscope/gq: a graph querying toolkit. *PVLDB*, 1(2):1404–1407, 2008.

[43] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[44] D. Z. Wang, E. Michelakis, M. Garofalakis, and J. M. Hellerstein. Bayesstore: managing large, uncertain data repositories with probabilistic graphical models. *Proc. VLDB Endow.*, 1(1):340–351, 2008.

[45] 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.*, 20(12):1669–1682, 2008.

[46] M. L. Yiu, N. Mamoulis, X. Dai, Y. Tao, and M. Vaitis. Efficient evaluation of probabilistic advanced spatial queries on existentially uncertain data. *IEEE TKDE*, 21(1), 2009.

[47] X. Zhang and J. Chomicki. On the semantics and evaluation of top-k queries in probabilistic databases. In *DBRank*, 2008.

# APPENDIX

## A. EXPECTED RELIABLE DISTANCE

The bound on the number of graphs needed to provide a good estimate for the Expected-Reliable-Distance problem is given by the following Lemma:

LEMMA 3 (EXPECTED-RELIABLE-DISTANCE). *Consider $\mathcal{G} = (V, E, P, W)$ with $n$ nodes. Consider accuracy parameters $\epsilon$ and $\delta$, 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 $E[d_i] = d_{\text{ER}}(s,t)$. By selecting $r \geq \frac{(n-1)^2}{2\epsilon^2} \ln(\frac{2}{\delta})$, we have*

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

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*

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

We move on to prove Lemma 3. We first reproduce the Chernoff bound:

THEOREM 1 (CHERNOFF BOUND). *Let $X_1, X_2, ..., X_r$ 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 $(\epsilon, \delta)$ approximation to $\mu$.*

We also reproduce the Hoeffding Inequality:

THEOREM 2 (HOEFFDING INEQUALITY). *Let $X_1, X_2, \ldots, X_r$ 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 $S = X_1 + ... + X_r$ we have*

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

We can now prove Lemma 3 for unweighted graphs:

PROOF. The first part of the lemma is a direct application of the Hoeffding inequality, Theorem 2. Observe that assuming connectivity, any distance in an unweighted 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. □

The number of samples is polynomial and not exponential as the brute-force algorithm. At first glance, this can still be prohibitively large due to the factor $(n-1)^2$. However, $(n-1)$ comes from an estimation of the largest possible path in the network. In real-world scenarios, networks have small diameter, due to the "small-world phenomenon", typically smaller than 20 (see , e.g., Figure 4(b)). Thus, in practice, the number of samples is much smaller. In order to bring Lemma 3 into practice, we introduce a threshold parameter $\rho$ for the reliability. We make the assumption that Expected-Reliable-Distance queries with connectivity below $\rho$ are not *interesting*. Together with accuracy parameters $\epsilon$ and $\delta$, the parameter $\rho$ is used to estimate the number of graphs that need to be sampled. In order to satisfy both parts of Lemma 3, we need to sample at least $r = \max\{\frac{3}{\epsilon^2\rho}, \frac{(n-1)^2}{2\epsilon^2}\} \cdot \ln(\frac{2}{\delta})$ graphs. Finally, in the case of weighted graphs, the bound needs to be adjusted by using the range of distances in the graph instead of the term $(n-1)$.

## B. CONDITIONAL PROBABILITIES

We next discuss how Lemma 1 can be applied when edges have conditional probabilities on their edges.

We relax the assumption of independence, as stated in the model definition in Section 2, by allowing edges to depend on one another. In particular, we assume that the probability of edge $e_1$ is conditioned on the existence of other edges, i.e., it is given by $p(e_1|e_2, e_3, \ldots)$. Construct graph $H$ from probabilistic graph $\mathcal{G}$ as follows: each edge of $\mathcal{G}$ is a node of $H$; each condition of the form $p(e_i|\ldots, e_j, \ldots)$ is a directed edge of $H$ with $e_j$ as the source and $e_i$ as the target. We now consider two cases for $H$, depending on its structure. Specifically, we consider whether $H$ is a directed acyclic graph (aka DAG) or not.

- $H$ **is a DAG**. Then, there exists a topological order $T$ of $H$'s nodes (i.e., $\mathcal{G}$'s edges). Now, sampling a random world is easy. We just need to sample each edge of $\mathcal{G}$ according to the topological order $T$. In other words, when the time comes to sample $e_i$ with $p(e_i|\ldots, e_j, \ldots)$, $e_j$ has already been sampled. Computing $T$ and sampling a world scales linearly to the number of edges in $H$ (i.e., the number of conditions in $\mathcal{G}$). Once we have a sample of $r$ independent worlds, we can directly apply Lemma 1.
- $H$ **is not a DAG**. In this case, obtaining an independent sample of $r$ graphs is more challenging. We employ Gibbs sampling, a Markov Chain Monte Carlo technique (MCMC). Beginning from an initial assignment of active and inactive edges, we randomly choose an edge $e$. We draw a sample from $e$'s distribution, based on the initial edge assignments. Edge $e$ may become active or inactive, regardless of its previous state. We repeat this process many times, each time choosing a random edge. It can be proven that this Markov chain yields, eventually, a sample (i.e., a world) from the edge-distribution (see [7] Ch.11 for details). Similar to the DAG case, once we obtain $r$ worlds, we can directly apply Lemma 1.

In brief, sampling a random world from probabilistic graphs that exhibit dependencies on their edges, may require sophisticated sampling techniques. However, Lemma 1 holds, as long as $r$ independent worlds have been sampled from $\mathcal{G}$. This discussion applies directly to Lemma 3. In this paper, we do not elaborate further on dependencies, since we are not aware of large real-world probabilistic graphs that exhibit them. However, we foresee such graphs's existence, and thus we believe that studying their properties is an important future direction.

# C. PROBABILISTIC RANDOM WALK

## C.1 An example of a walk

For additional intuition on the Probabilistic-Random-Walk, consider this example: assume that a drifter is in Boston and that there are three roads that she can take, one to New York, another one to Toronto, and one to Montreal. Each road has a proximity value indicating the inverse of the time it takes to cross it. Also, each road is labeled with 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. The roads to Toronto and Montreal are open, while the road to New York is closed. The drifter favors short roads so she chooses between the two roads, with probability relative to their proximity to Boston. If all roads were closed she would stay another night and wait for better weather the next day.

## C.2 Random walk transformation

The following theorem provides an equivalent definition of the walk (defined in Section 3.2) by taking advantage of the memoryless property inherent in this process. The walk on $\mathcal{G}$ can be transformed into a standard random walk process on a non-probabilistic graph $\overline{G} = (V, \overline{E}, \overline{W})$.

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 $\overline{G}(V, \overline{E}, \overline{W})$, where $\overline{E} = E \cup S$, with $S = \{(u, u)\}$ (i.e., the set of self-looping edges). $\overline{W} = \{\overline{w}(u, v)\}$, with*

$$\overline{w}(u,u) = \prod_{(u,q)\in E} (1 - p(u,q)), \text{ and}$$

$$\overline{w}(u,v) = \sum_{G|(u,v)\in G} \frac{w(u,v)}{\sum_{(u,q)\in G} w(u,q)} \Pr[G] \quad (1)$$

A direct corollary of Theorem 3 is that the stationary distribution of Probabilistic-Random-Walk on $\mathcal{G}$ is the same as the one of the standard random walk on $\overline{G} = (V, \overline{E}, \overline{W})$. Thus, once we have computed $\overline{G} = (V, \overline{E}, \overline{W})$ we can apply standard algebraic techniques to solve the stationary distribution problem and compute any Random-Walk-Distance distance. The complexity of computing each weight $\overline{w}(u, v)$ 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 $\overline{w}(u, v)$ becomes an intractable problem.

We propose a Monte Carlo algorithm for computing the weights $\overline{w}(u, v)$. We sample different outgoing edges, for each node $u$, and estimate Equation (1) by taking the sum of probabilities over the sampled graphs only, instead of using all possible graphs. The Chernoff bound can be applied again to show that a small number of samples per node $u$ is sufficient to approximate the weights $\overline{w}(u, v)$.

## C.3 Transformation with grouping

We begin our discussion by considering a special case:

**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 Erdős-Renyi graph 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:

$$\overline{w}(u,u) = (1-p)^{d_u}, \text{ and}$$

$$\overline{w}(u,v) = \sum_{k=1}^{d_u} \frac{\binom{d_u}{k}}{k} p^k (1-p)^{d_u - k} = \frac{1 - \overline{w}(u,u)}{d_u},$$

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 outgoing 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 $i$, $1 \le i \le R$. Also let $q_i$ be the probability of the edges in group $i$. Omitting some simple algebra, the equations for the weights now become:

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

where $\overline{w}_{u,i}$ denotes the weights on all outgoing edges to nodes of the group $i$ (note that because of symmetry they have all the same weight). The function $C(i, m_1, .., m_R)$ 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:

$$C(i, m_1, .., m_R) = \binom{n_1}{m_1} \cdot .. \cdot \binom{n_i - 1}{m_i} \cdot .. \cdot \binom{n_G}{m_G}.$$

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

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

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

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

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.

## C.4 Random walk $k$-NN

The answer to the random walk $k$-NN problem is the set of $k$ nodes that have the largest stationary distribution values. In order to compute the $k$-NN results for a source node $s$, we propose to simulate the random walk and approximate

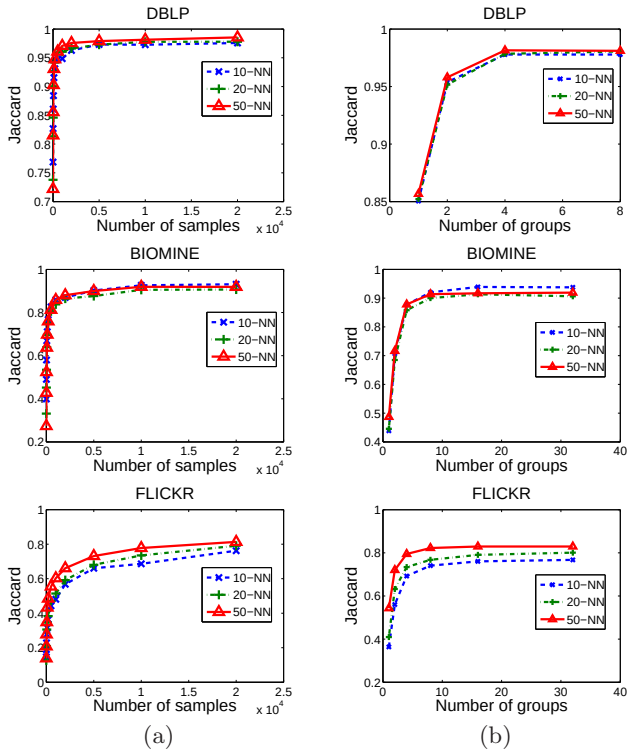



**Figure 7: Performance of $k$-NN vs. (a) number of samples, (b) number of groups.**

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 [6] 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 (the transformed graph) $\overline{G}$ instead of $\mathcal{G}$ using Theorem 3 from Section C.2. This way, we drastically reduce the amount of randomness at each step of the walk (i.e., we save the time needed to check if each outgoing edge of the current node is active). We note that any technique for personalized or individual PageRank computation on deterministic graphs, e.g., [16] can be directly applied to $\overline{G}$.

## C.5 Transformation efficiency

Theorem 3 shows the equivalence of the Probabilistic-Random-Walk to a random walk on a deterministic weighted graph. The direct computation of the transformed graph $\overline{G}$ can be performed in a per node basis. However, it scales exponentially to the number of the node's outgoing edges, making it practically intractable to compute exactly for out-degrees greater than 30. Thus, we implemented the sampling algorithm for the transformation (with an option to group edges). We also implemented the $k$-NN algorithm from Appendix C.4.

In Figure 7(a) we present the performance of the transformation in terms of success for the $k$-NN query. Success in the $k$-NN query was computed using Jaccard's coefficient for the $k$-NN sets of our method and the true $k$-NN. We consider the true $k$-NN to be the result using 50K samples for the transformation, after observing empirically that the results were stable for that number of samples.

**Table 2: Grouping Speedup.**

| *Median, 200 Worlds* | | | | | |
|---|---|---|---|---|---|
| Number of groups | 2 | 4 | 8 | 16 | 32 |
| DBLP | 1.02 | 1.00 | 1.00 | 1.00 | 1.00 |
| BIOMINE | 1.20 | 1.15 | 1.10 | 1.06 | 1.03 |
| FLICKR | 1.33 | 1.32 | 1.30 | 1.28 | 1.24 |

The transformation scales linearly to the number of samples and it can take a few minutes (for 1000 samples) to a few hours (for 50K samples) for BIOMINE, 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 for the Random-Walk-Distance we performed $1M$ random walks per experiment after empirically observing that this number was large enough. The teleport probability was set to 0.20. Notice that less than 1K samples in DBLP and 10K in BIOMINE yielded more than 90% accuracy. In FLICKR which is a more volatile graph since it is very dense and has edges of extremely low probability, the performance is around 80% at 50K samples and we need to sample 200K worlds to reach 90% performance.

In Appendix C.3 we presented a grouping heuristic for the graph transformation of Section C.2. We performed 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 7(b). As expected DBLP converges very fast (4 groups are enough). Recall from Table 3 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 grouping technique. We note that for this experiment we used everywhere 20K MC samples.

We also present wall-clock speedups in Table 2. As expected, efficiency increases as the number of groups decrease. The gains, however, are overall moderate due to the power law out-degree property of our datasets. In particular, only the nodes that have larger out-degree than the number of groups are affected from grouping. For instance, less than 5% of the nodes in BIOMINE have degree more than 16. Consequently on 95% of the nodes the grouping with 16 groups has no effect. At the same time, more than 50% of the savings come from nodes of out-degree greater than 100, which comprise less than 1% of the total number of nodes. In absolute numbers, a complete transformation of FLICKR with 16 groups took approximately 1000 seconds, instead of 1300.

# D. DATASETS

The probabilities for DBLP and FLICKR have been computed from real information based on simple probability assignment models. We choose to use simple models, since model selection, in general, is beyond the scope of this pa-

**Table 3: Datasets' characteristics.**

| Dataset | $\|V\|$ | $\|E\|$ | Max Outdegree |
|---|---|---|---|
| DBLP | 226K | 1.4M | 238 |
| BIOMINE | 1M | 10M | 139603 |
| FLICKR | 77K | 20M | 5765 |

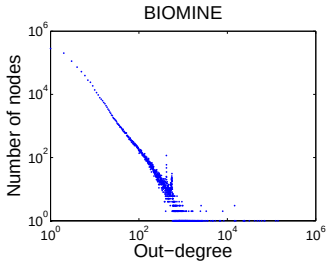



**Figure 8: Out-degree distribution of BIOMINE.**

**Table 4: Frequency of infinite distance values.**

| | BIOMINE | DBLP | FLICKR |
|---|---|---|---|
| MAJORITY | 0.83 | 0.78 | 0.42 |
| EXPECTEDREL | 0.69 | 0.56 | 0.35 |
| MEDIAN | 0.69 | 0.56 | 0.35 |

per. BIOMINE is already labeled with probabilities. All the probabilistic graphs are connected, but obviously many disconnected worlds can be instantiated and thus infinite distances can occur. Table 3 summarizes size and maximum out-degree of the datasets.

The datasets follow a power-law out-degree distribution, commonly found in scale-free networks. We present the degree distribution of BIOMINE in Figure 8 as an example and note that the others are similar. Observe that there are some central nodes, connected to 5% of the database.

## E. INFINITE DISTANCE

Table 4 is complementary to Figure 4(b). There are many infinite distances in our datasets. For example, for 56% of the pairs of nodes, the median distance is infinite. Recall from Figure 8 that there are many nodes with one or two edges. Also recall from Figure 4(a) 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.