Exploration of Monte-Carlo based Probabilistic Query Processing in Uncertain Graphs

Tobias Emrich, Hans-Peter Kriegel, Johannes Niedermayer, Matthias Renz,
André Suhartha, Andreas Züfle
Institute for Informatics, Ludwig-Maximilians-Universität München
Oettingenstr. 67, D-80538 München, Germany
{emrich,kriegel,kroeger,niedermayer,renz,suhartha,zuefle}@dbs.ifi.lmu.de

ABSTRACT

This demo presents a framework for running probabilistic graph queries on uncertain graphs and visualizing their results. The framework supports the most common uncertainty model for uncertain graphs, i.e. existential uncertainty for the edges of the graph. A large variety of meaningful graph queries are supported, such as shortest path, range, kNN, reverse kNN, reachability and various aggregation queries. Since the problem of exact probability computation according to possible world semantics is in $\#P$-Time for many combinations of model and query, and since ignoring uncertainty (e.g. by using expectations only) will yield counterintuitive and hard to interpret results, our framework uses an optimized version of Monte-Carlo sampling to estimate the results which allows us not only to perform queries that conform to *possible world semantics* but also to sample only parts of a graph relevant for a given query. The main strength of this framework is the visualization combined with statistic hypothesis tests, which gives the user not only the estimated result of a query, but also an indication of how significant and reliable these results are. The aim of this demonstration is to give an intuition that a sampling based approach to probabilistic graphs is viable, and that the estimated results quickly converge even for very large graphs. A video demonstrating our framework can be downloaded at *http://www.dbs.ifi.lmu.de/Publikationen/videos/PGraph.html*

Categories and Subject Descriptors

G.2.2 [**DISCRETE MATHEMATICS**]: Graph Theory

Keywords

Monte-Carlo, Sampling, Probabilistic Graph, Visualization

1. INTRODUCTION

Uncertainty is ubiquitous in the physical world. It can be introduced by faulty measurements e.g. in wireless sensor networks, but also by privacy enhancing transformations of collected data. Therefore, in the last years database researchers developed tools, data structures and query algorithms for uncertain data, both for relational databases, but also for spatio-temporal and multimedia databases [1].

Spatial networks, such as street networks and information routing networks, but also social networks, the World Wide Web and even biological networks, e.g.protein-protein interaction networks [2] can be modeled by graphs. Within many of these applications, uncertainty is a major concern. Routers can loose connectivity due to overload or physical damage, congestion can obstruct streets, and the information given by users in social networks might be erroneous.

However, although addressing uncertainty in databases is of critical interest both for researchers and industry, the problem of exact probability computation according to possible world semantics is in $\#P$-Time for nearly all combinations of model and query. Discarding uncertainty information often makes results counterintuitive and difficult to interpret such that approximation of the results achieved by possible world semantics is usually a better choice. To provide an insight into the topic of querying uncertain graphs under the possible worlds semantics, we developed a tool for visualizing sampling-based queries on probabilistic graph data. In accordance to related research [2], our graph model assumes nodes to be certain while edges own existential uncertainty, i.e. edges either exist or do not exist, while the length of an edge is assumed to be fixed. Based on a simple interface, the user can create graphs, load them from an XML file and modify the graphs by changing edge weights and probabilities. He can perform queries such as kNN queries[2], RkNN [3] queries or finding reachable nodes, and he can retrieve statistics over the probabilistic graph, such as its diameter or its largest connected component. Since queries in probabilistic graphs are inherently uncertain, the user receives probability distributions over possible results or probabilities of objects being part of the result. Graph data and resulting probabilities are visualized by employing one of three visualization techniques.

With our demonstration we aim at providing an insight

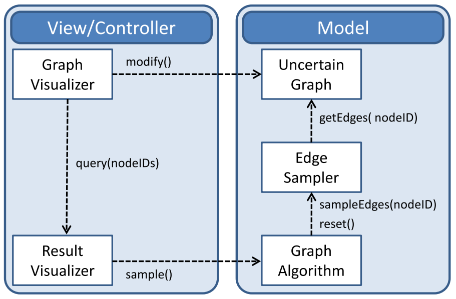


Figure 1: High Level Framework

into the sampling process, answering questions such as *How many samples are sufficient to achieve high accuracy of the query result?*. We approach this problem both from a visualization perspective and by employing statistical models. visualization allows the user to find out when the sampling process converges to a statistical distribution. The implemented statistical models draw the same conclusions in a formal manner.

The remainder of this paper is organized as follows. In Section 2 we introduce the theoretical foundations of our demonstration. Section 3 introduces the building blocks of our framework, namely queries and visualizations, and provides an insight into implementation-specific details. In Section 4 we give an insight into the aspects that we will demonstrate during our presentation.

2. THEORETICAL BACKGROUND

Before introducing our framework, let us first provide a short overview over the theoretical background of our work. One of the most basic concepts addressed within our work is the concept of a *Graph*.

Definition 1 (Graph). *An (undirected) weighted graph G is a tuple $G = (V, E, w)$ with V being a set of vertices (nodes), E being a set of edges, $E \subseteq \{(v_1, v_2)|v_1, v_2 \in V\}$, and $w : E \to \mathbb{R}$ being a weighting function.*

Within our setting, the set of edges is not fixed, but each edge has a probability of existence, while the existence of one edge does not affect the existence of another edge:

Definition 2 (Probabilistic Graph). *A probabilistic graph under existential uncertainty is an (undirected) weighted graph $P = (V, E, w, p)$, with $p : E \to [0, 1]$ being a function that returns for each $e \in E$ the probability of being an edge of G.*

A realization of P can be drawn by applying the function p to each edge $e \in E$, resulting in a graph $G = (V, E' \subseteq E, w)$. Each of these realization corresponds to a *possible world* of P. The number of possible worlds of P is usually exponential in the number of uncertain edges, i.e. in the number of edges $e \in E$ with $0 < p(e) < 1$. Therefore materializing every possible world of P, computing its probability, and computing results exhaustively is not a matter of choice. Sampling is usually much faster and can achieve accurate results as well.

One of the most important questions when employing sampling is to find out how many samples are sufficient to find the correct result with high probability. This problem was one of the driving factors for our demonstration. From a statistics point of view, *hypothesis tests* serve this purpose. In our framework, we employed the binomial test statistic. With this technique it is possible to determine the probability (*confidence*) that a given result, retrieved by sampling, will deviate by more than a predefined threshold (the *confidence interval*).

Visual tools developed for gaining insight into the sampling process will be introduced in the following section.

3. FRAMEWORK DESCRIPTION

Framework Overview. The developed framework has been built with extensibility in mind such that new queries and corresponding result visualizations can be easily integrated. A simplified overview is given in Figure 1. The main part within the framework is of course the *uncertain graph* itself. It is visualized and can be modified by the *graph visualizer*. Additionally the *graph visualizer*, which is the main graphical interface, lets the user choose an algorithm, set the input parameters and start the sampling for the specified *graph algorithm* via the corresponding visualizer. Each *graph algorithm* may only access edges of the *uncertain graph* using the *edge sampler*. The *edge sampler* on the other hand requests the edges and connected nodes from the *uncertain graph* and uses Monte-Carlo sampling to create a possible world for these edges. The *edge sampler* has to memorize which edges that have already been sampled during one run (resulting in one sample result) of the algorithm and is reset when the algorithm terminates the current run. Each *graph algorithm* is controlled by a corresponding *result vizualizer* which displays the accumulated result after each sample run of the algorithm.

Supported Queries. A wide variety of queries from traditional database research can be applied to graph structures. Furthermore, mathematical statistics over a graph such as its diameter can provide useful insights into a particular topic. Due to space constraints, we only focus on an informal definition of each query:

- kNN query: Given a query object Q and several objects located on arbitrary nodes within the network, a kNN query returns the k objects in the graph closest to Q[4].[1] In our probabilistic setting, this query returns, for each object, the probability of being in the kNN set of object Q . The query can, for example, be employed in spatial networks to determine the gasoline stations closest to a car.
- RkNN query: The graph RkNN query, defined in [3], is the converse of a graph kNN query. Given a query object Q, we aim at determining the objects that have Q as one of their k nearest neighbors. Again, this query returns, for each object, the probability of being in the RkNN set of Q . A useful application of this query is outlier detection.
- Shortest-Path-Query: This query computes the length of the shortest path between two pre-defined nodes in the network. In a probabilistic setting, the result is not a single value but instead a distribution of the length of shortest paths in all possible worlds.
- Connected Components: Our framework allows to compute the number of connected components within the graph and the size of its largest connected component. Furthermore, after defining a starting node, one can compute the size of the connected component containing the starting node.
- Diameter: The diameter of a graph is defined as the maximum length of a shortest path between two arbitrary nodes within the network. Our framework allows to compute the diameter of a probabilistic graph as the largest non-infinite distance between any two nodes in the network.

Visualizations. Evaluated graphs and the corresponding query results can be visually analyzed with a wide variety of tools. After creating a graph, the user has access to a Node-Edge-Visualization of the graph, see Figure 2. The

[1]For the sake of simplicity, in contrast to [4] we only allow objects to be located on nodes of the graph, not edges.

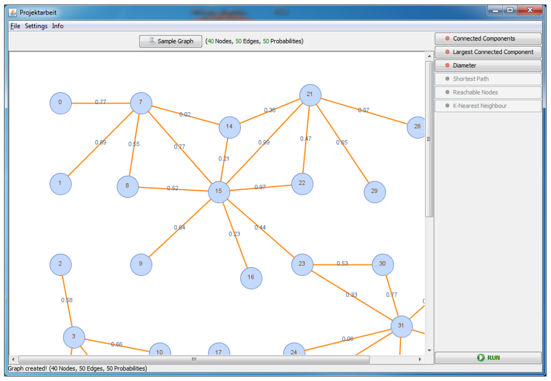


Figure 2: The Graph Visualizer

user can decide whether edge weights should show the length of an edge or its probability of existence, or whether object IDs or vertex IDs are shown. If the aspired query expects a start/end node as an input, the user can select them by left-clicking. Objects for the kNN and RkNN query can be positioned and deleted in a dedicated edit mode.

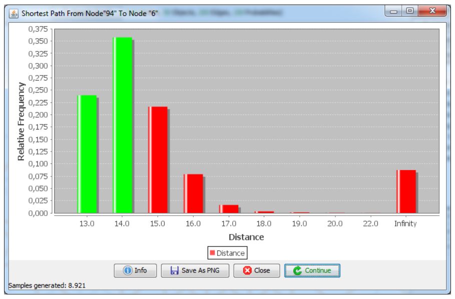
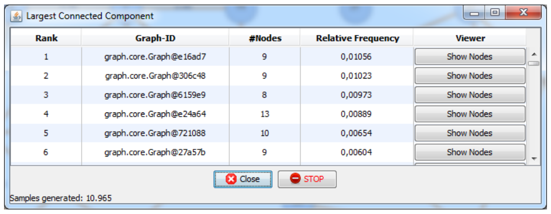


Figure 3: Probability Count Histogram

After performing a query resulting in a real-numbered result, such as the Shortest-Path-Query, the number of connected components or the diameter of a graph, a user can access the probability distribution of the result (see Figure 3). To allow the user an insight into the convergence of a query, this probability distribution is shown directly after drawing the first sample and updated continuously whenever new samples are drawn. This allows the user to gain insight into the confidence of a result at a given number of samples drawn from the graph.



Largest Connected Component

Rank	Graph-ID	#Nodes	Relative Frequency	Viewer
1	graph.core.Graph@e16ad7	9	0,01056	Show Nodes
2	graph.core.Graph@306c48	9	0,01023	Show Nodes
3	graph.core.Graph@6159e9	8	0,00973	Show Nodes
4	graph.core.Graph@e24a64	13	0,00889	Show Nodes
5	graph.core.Graph@721088	10	0,00654	Show Nodes
6	graph.core.Graph@27a57b	9	0,00604	Show Nodes

Samples generated: 10.965

Figure 4: The Subgraph Visualization

Whenever a user performs a query that returns a set of objects, such as the (R)kNN query, a ranking is shown that shows, for each object, the probability of being within the result set. Again, this result is updated continuously when drawing samples from the graph. Figure 4 shows the result of a largest-connected-component query.

Key Features. We want to highlight three key features of the system which yield more efficiency on the on hand and more intuitive understanding of the probabilistic results for the user on the other hand.

Realtime-result-visualization means that during query processing the result of each sample run is directly integrated into the overall sampling result. As a consequence each sample run updates the result visualization and the user can intuitively determine when certain probabilistic results converge (meaning that the confidence of the current sampling result being close to the exact result is very high).

On-the-fly-sampling [2] is a technique which is enforced by the architecture of the framework. Each integrated algorithm is required to only access the uncertain graph using the sampler component which then samples the edges of a node. In doing so, it is assured that each algorithm only samples the part of the graph which is necessary for computing the result. E.g. a shortest-path query between two nearby nodes does not require to sample the whole graph, which would otherwise yield a large computational overhead.

Integrated statistical tests give the user an additional assistance when to stop the monte-carlo sampling procedure. The tests are performed after each sample run of an algorithm and are integrated into the result visualization.

4. PRESENTATION

During the demonstration we will describe some crucial applications which require to represent the underlying model as an uncertain graph and discuss the main motivations for this system. We will show how synthetic and real-world uncertain graphs can be created, loaded or modified into the system. A major focus of the demonstration will lie on performing different graph algorithms on the uncertain graphs. We will show that the online-sampling approach yields an interactive and intuitive way for the user to know when to stop the query processing. Additionally we will present how the decision to stop an algorithm can be facilitated by integrated statistical tests. Using the on-the-fly sampling, which is enforced for each algorithm in the framework it will be demonstrated, that results converge very fast even for extremely large graphs, since only relevant parts of the graph are sampled. Last but not least we will describe the different result visualization techniques of the system and how queries and corresponding visualizations can be modularly integrated into the system.

5. ACKNOWLEDGEMENTS

Part of this work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant number KR 3358/4-1.

6. REFERENCES

[1] C. C. Aggarwal and P. S. Yu, "A survey of uncertain data algorithms and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 5, pp. 609–623, 2009.

[2] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, "k-nearest neighbors in uncertain graphs," *PVLDB*, vol. 3, no. 1, pp. 997–1008, 2010.

[3] M. L. Yiu, D. Papadias, N. Mamoulis, and Y. Tao, "Reverse nearest neighbors in large graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 4, pp. 540–553, 2006.

[4] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in *VLDB*, 2003, pp. 802–813.