BMC: An Efficient Method to Evaluate Probabilistic Reachability Queries

Ke Zhu, Wenjie Zhang, Gaoping Zhu, Ying Zhang, and Xuemin Lin

University of New South Wales, Sydney, NSW, Australia
{kez,zhangw,gzhu,yingz,lxue}@cse.unsw.edu.au

Abstract. Reachability query is a fundamental problem in graph databases. It answers whether or not there exists a path between a source vertex and a destination vertex and is widely used in various applications including road networks, social networks, world wide web and bioinformatics. In some emerging important applications, uncertainties may be inherent in the graphs. For instance, each edge in a graph could be associated with a probability to appear. In this paper, we study the reachability problem over such uncertain graphs in a threshold fashion, namely, to determine if a source vertex could reach a destination vertex with probabilty larger than a user specified probability value t. Finding reachability on uncertain graphs has been proved to be NP-Hard. We first propose novel and effective bounding techniques to obtain the upper bound of reachability probability between the source and destination. If the upper bound fails to prune the query, efficient dynamic Monte Carlo simulation technqiues will be applied to answer the probabilitistic reachability query with an accuracy guarantee. Extensive experiments over real and synthetic datasets are conducted to demonstrate the efficiency and effectiveness of our techniques.

1 Introduction

In many real world applications, complicatedly structured data could be represented by graphs. These applications include Bioinformatics, Social Networks, World Wide Web, etc. Reachability query is one of the fundamental graph problems. A reachability query answers whether a vertex u could reach another vertex v in a graph. Database community has put considerable efforts into studying the reachability problem, for example, [7], [5], [1], [14], [9], [2], [8], [4], [3], etc.

All of the above works focus on the applications where edges between two vertices exist for certain. However, in many novel applications, such an assumption may not capture the precise semantics and thus the results produced are also imprecise.

Example 1: In Protein-Protein interaction networks, an edge between two proteins means they have been observed to interact with each other in some experiments. However, not all interactions can be consistantly observed in every experiment. Therefore, it is more accurate to assign probabilities to edges to represent the confidence on the relationship. In this application, biologists may want to query whether a particular protein is related to another protein through a series of interactions.

J.X. Yu, M.H. Kim, and R. Unland (Eds.): DASFAA 2011, Part I, LNCS 6587, pp. 434–449, 2011.

Example 2: Social Network Analysis has recently gained great research attention with the emergence of large-scale social networks like LinkedIn, Facebook, Twitter and MySpace. In these social networks, connections between entities/individuals(vertices) may not be completely precise due to various reasons including errors incurred in data collection process, privacy protection, complexed semantics, disguised information, etc([22]).

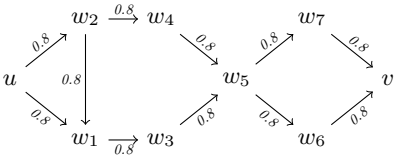


Fig. 1. A running example

In above applications, an edge connecting two vertices is associated with a probability value indicating the confidence of its existence. Reachability queries over this kind of uncertain graphs are thus called Probabilistic Reachability Queries. The Probabilistic Reachability problem is intrinsically difficult. As a running example in Fig. 1, this graph consists of only 11 edges. To accurately answer the Probabilistic Reachability from u to v, we need to enumerate up to 2^{11} possible instances of the uncertain graph. For each of these instances, we need to check whether u can reach v, and then aggregate the probabilities of the instances in which u can reach v. In [10], Valiant has proved this problem is NP-Hard.

Monte Carlo simulation provides an approximate solution to this problem. A considerable number of articles([15],[16],[17],[18],[19], etc) studied how to use Monte Carlo simulation to solve the probabilistic reachability problem. The focus of those studies are on utilizing different sampling plans to reduce sampling error. Due to the dramatical increase of the scale of graphs and the large number of iterations required by Monte Carlo simulation to guarantee the accuracy, the computational cost of traditional Monte Carlo method is still considerably expensive.

In this paper, we propose a more efficient dynamic Monte Carlo method to approximate the answer. This dymanic Monte Carlo method will only simulate necessary part of the graph and will share most of the overlapping cost between different iterations. In addition to that, we also propose an index which can assist in calculating upper bound of probabilistic reachability. Queries pruned by the bound do no need to be approximated by the Morte Carlo method which is relatively more expensive. The main contributions of the paper are:

1. To the best of our knowledge, we are the first to address the efficiency issues of Probabilistic Reachability Queryby using indexing techniques. We formally define Probabilistic Reachability Queryusing *Possible World* semantics.
2. We propose an efficient dynamic Monte Carlo algorithm to calculate approximate result. In addition, we also give a theoretical accuracy guarantee for the Monte Carlo method.
3. We propose an index which efficiently calculates the upper bound of Probabilistic Reachability Queries.
4. We perform extensive experiments on real datasets and synthetic datasets to demonstrate the efficiency of our proposed method.

Table 1. Notations

$u \rightsquigarrow v$, $u \not\rightsquigarrow v$	u can reaches v; u cannot reach v
$p(e)$	the probability that edge e will exist
$R_{u,v}$	the probability that u could reach v
$R^U_{u,v}$	the upper bound probability that u could reach v
ω, Ω	a possible world and the set of all possible worlds respectively
p_ω	the probability of a possible world
$s(u,v)$	the shortest distance between u and v
$Prob(Event)$	the probability that an *Event* will occur

The whole paper is organized as follows: Section 2 will introduce the background knowledge of this problem. Section 3 will briefly outline our Bound and Monte Carlo(BMC) framework. Section 4 will propose a novel bound-based scheme to address the problem. Section 5 will introduce our dynamic Monte Carlo method. Section 6 will demonstrate and analyze the experiments. Section 7 will introduce related works and Section 8 concludes the paper.

2 Background

2.1 Problem Definition

In this paper, we study the reachability problem in graphs in which each edge is associated with an existence probability and we call such graphs *Uncertain Graphs*

Definition 1 (Uncertain Graph). *An uncertain graph is defined as* $G = (V, E, P_E)$ *where* V *is the set of vertices,* E *is the set of edges and* $P_E : E \rightarrow (0, 1]$ *is the edge probability function. We use* $p(e)$ *to denote the probability that* e *exists where* $e \in E$*.*

The probability that vertex u could reach v can be calculated by summing the probability of all possible combinations of the edge states. Each of the combination corresponds to a *Possible World* in the *Possible World* semantics. We use $R_{u,v}$ to represent the probability of being reachable and $\bar{R}_{u,v}$ otherwise.

Definition 2 (Possible World). *Let* $x_e = 1$ *if* e *exists and* $x_e = 0$ *if otherwise. We call* ω *a possible world where* $\omega = \{x_e | e \in E\}$*.*

We use Ω to denote the set of all possible worlds of an uncertain graph and let $r_{u,v}(\omega) = 1$ if u could reach v in ω or $r_{u,v}(\omega) = 0$ when otherwise. We also use p_ω to represent the probability for ω to occur. $p_\omega = \prod_{e \in E} h(e)$ where:

$$h(e) = \begin{cases} p(e) \text{ if } x_e = 1 \\ 1\text{- } p(e) \text{ if } x_e = 0 \end{cases}$$

Definition 3 (Probabilistic Reachability). *The probabilistic reachability between two vertices* u *and* v*,* $R_{u,v}$*, is the sum of probability of all possible worlds in which* u *can reach* v*. That is,* $R_{u,v} = \sum_{\omega \in \Omega} r_{u,v}(\omega) \cdot p_\omega$*.*

Definition 4 (Probabilistic Reachability Queries). *Given a large uncertain database graph G, two vertices u, v, where $u, v \in V$, and a threshold t where $1 \geq t > 0$, the database outputs true if $R_{u,v} \geq t$ or false if $R_{u,v} < t$. We call this type of queries* Probabilistic Reachability Queries.

2.2 Preliminaries

Naive Enumeration. Without any pruning strategy, we need to enumerate every possible world $\omega \in \Omega$ and to increment the probability of success or failure till t or $1 - t$ is reached. Algorithm **NaiveEnumerate** outlines the naive enumeration process.

Procedure. `NaiveEnumerate`(G, u, v, t)

```
1  begin
2      fail, operate = 0 ;
3      Ω = all possible worlds of G ;
4      foreach ω in Ω do
5          if u can reach v in ω then
6              operate = operate + p_ω ;
7          else
8              fail = fail + p_ω ;
9          if fail > 1 - t then
10             return false ;
11         if operate ≥ t then
12             return false ;
13 end
```

Monte Carlo Sampling. The complexity of caculating probablistic reachability has been proved to be NP-Hard[10]. The cost grows exponentially as size of graphs grows. Monte Carlo sampling method is generally a widely accepted method of approximating the result. Briefly, it has three steps:

1. Randomly and independently determine a state for every edge in the graph according to the operational probability of each edge. A sample graph consists of all edges with a *exist* state.
2. Test the the reachability for this sample graph.
3. Repeat the above step 1 and 2 for k iterations.

The approximate probabilistic reachability is: $R'_{u,v} = \frac{\sum_{\omega \in \Omega_k} r_{u,v}(\omega)}{k}$ where Ω_k is a set of k sampled states.

3 Framework

As finding the probablistic reachability is infeasible when data graphs are large, we propose a framework integrating an effective bounding-pruning technique and an efficient Monte Carlo sampling technique. Intuitively, if $R_{u,v}$ is small and the threshold is large, it is possible to obtain an upper bound, $R^U{}_{u,v}$, to immediately

reject the query. If the bounding technique fails to prune the query, then Monte Carlo sampling technique will be applied to produce an approximate answer. We observe that a major portion of the traditional Monte Carlo simulation can be shared, thus we propose a more efficient Dynamic Monte Carlo method to approximate the result. The following sumarizes the major steps in our framework.

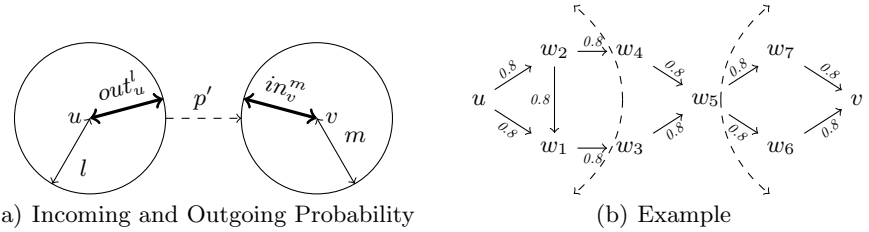
1. We create an index on the database graph so that the upper bound of the exact reachability could be calculated efficiently. We will attempt to prune the query by calculating the upper bound of the probablistic reachability. This technique is to be detailed in **Section 4**.
2. If the upper bound cannot prune the query, we will sample a number of possible worlds and use our proposed Dynamic Monte Carlo simulation to estimate the reachability. This technique is to be detailed in **Section 5**.

4 Upper Bound Index

Naive Enumeration is impractical to answer Probabilistic Reachability Queries. However, Probabilistic Reachability Queries only need to answer whether the reachability is above a threshold t, rather than the exact probabilistic reachability. With the help of indices, we can efficiently calculate an effective upper bound of the reachability between the source and the destination. If the upper bound can be used to prune the query before Monte Carlo Simulation, we can avoid the relatively more expensive sampling and reachability testing.

As mentioned previously, it is infeasible to enumerate every single possible world when a graph is large. The following observations inspired us to propose an efficient upper bound index:

Observation 1: Many real-world graphs are sparse and a local neighbourhood graph surrounding a vertex is usually small in sparse graphs. It is affordable to enumerate all possible worlds in a small neighbourhood graph.
Observation 2: The local neighbourhood structure surrounding a vertex can usually provide an upper bound of its ability to reach(or to be reached by) other vertices.



(a) Incoming and Outgoing Probability (b) Example

Fig. 2. Upper Bound Calculation

As shown in Fig. 2 (a), a vertex u has to reach at least one vertex outside the circle of radius l before it can reach vertex v if l is less than the unweighted shortest distance between u and v. We will use $s(u, v)$ to denote the shortest distance between u and v. Similarly, a vertex v has to be reached by at least

one vertex outside the circle of radius m before it can be reached by v if m is less than $s(u, v)$. The probabilistic reachability is bounded from above by the *outgoing probability* and the *incoming probability*, which are defined as below.

Definition 5 (Outgoing and Incoming Probability). *The outgoing probability of vertex u, ${out^k}_u$, represent the probability that u could reach at least one vertex w where $s(u, w) \geq k$. Similarly, the incoming probability of vertex v, ${in^k}_v$, means the probability that at least one vertex w could reach v where $s(w, v) \geq k$.*

Table 2. u's Outgoing Probability

Possible World ω	p_ω	Possible World ω	p_ω
$\{uw_1, \overline{uw_2}, w_2w_1, w_2w_4, w_1w_3\}$	0.08192	$\{uw_1, \overline{uw_2}, \overline{w_2w_1}, w_2w_4, w_1w_3\}$	0.02048
$\{uw_1, \overline{uw_2}, w_2w_1, \overline{w_2w_4}, w_1w_3\}$	0.02048	$\{uw_1, \overline{uw_2}, \overline{w_2w_1}, \overline{w_2w_4}, w_1w_3\}$	0.00512
$\{\overline{uw_1}, uw_2, w_2w_1, w_2w_4, w_1w_3\}$	0.08192	$\{\overline{uw_1}, uw_2, w_2w_1, \overline{w_2w_4}, w_1w_3\}$	0.02048
$\{\overline{uw_1}, uw_2, w_2w_1, w_2w_4, \overline{w_1w_3}\}$	0.02048	$\{\overline{uw_1}, uw_2, \overline{w_2w_1}, w_2w_4, \overline{w_1w_3}\}$	0.00512
$\{\overline{uw_1}, uw_2, \overline{w_2w_1}, w_2w_4, w_1w_3\}$	0.02048	$\{uw_1, uw_2, w_2w_1, w_2w_4, w_1w_3\}$	0.32768
$\{uw_1, uw_2, w_2w_1, \overline{w_2w_4}, w_1w_3\}$	0.08192	$\{uw_1, uw_2, w_2w_1, w_2w_4, \overline{w_1w_3}\}$	0.08192
$\{uw_1, uw_2, \overline{w_2w_1}, w_2w_4, w_1w_3\}$	0.08192	$\{uw_1, uw_2, \overline{w_2w_1}, \overline{w_2w_4}, w_1w_3\}$	0.02048
$\{uw_1, uw_2, \overline{w_2w_1}, w_2w_4, \overline{w_1w_3}\}$	0.02048		

In Table 2, we give an example of how to calculate the outgoing probability of u when the radius is 2 for the graph in Fig. 2 (b). In this example, we list all possible worlds in which u can reach at least one of w_3 and w_4. Each possible world is represented by a list of edge states. For example, $\{uw_1, \overline{uw_2}\}$ represents a possible world in which the edge between u and w_1 exists, and the edge between u and w_2 does not exist. The outgoing reachability is the aggregated probability of all listed possible worlds. Please note that we do not need to enumerate edges which are further than the specified radius because they have no effect on the outgoing reachability.

Generally speaking, as the outgoing(or incoming) edges get denser, it is more likely to reach(or to be reached by) other vertices. We can index the outgoing and incoming probability for every vertex for a specific small radius. In addition to the incoming and the outgoing probability, every u-v cut will bound the reachability between u and v from above.

Definition 6 (u-v Cut and Non-overlapping u-v Cut Set). *A u-v cut is a set of edges which will make v unreachable from u if all edges in the set are missing. A non-overlapping set of u-v cuts $C^{l,m}_{u,v}$ is a set of u-v cuts such that $\forall c_1, c_2 \in C^{l,m}_{u,v}, c_1 \cap c_2 = \emptyset$ and $\forall c \in C^{l,m}_{u,v}, \forall (x, y) \in c, s(u, y) > l \wedge s(x, v) > m$.*

Let us denote a cut with c, the event that all edges in c are missing with $Cut(c)$ and the probability for this event to occur with $Prob(Cut(c))$. For any l and m where $l + m \leq s(u, v)$ and we are given $C^{l,m}_{u,v}$, u could not reach v if any one of the following conditions is true: 1). $\forall x \in V, \neg(s(u, x) \geq l \wedge u \rightsquigarrow x)$. 2). $\exists c \in C^{l,m}_{u,v}$, $Cut(c)$ occurs. 3). $\forall x \in V, \neg(s(x, v) \geq m \wedge x \rightsquigarrow v)$

Theorem 1. *For any u and v, if we are given ${out^l}_u$, ${in^m}_v$, $C^{l,m}_{u,v}$ where $l + m \leq s(u, v)$, l ,$m \geq 0$:*

$$R_{u,v} \leq out_u^l \cdot in_v^m \cdot \prod_{c \in C_{u,v}^{l,m}} 1 - Prob(Cut(c)). \quad (1)$$

Proof. We will prove the above theorem by proving the following three conditions are necessary conditions and they are independent to each other.

1. If $\forall x \in V, \neg(s(u,x) \geq l \wedge u \rightsquigarrow x)$, then $(u \not\rightsquigarrow v)$ because $s(u,v) \geq l$. Therefore if $u \rightsquigarrow v$, then $\exists x \in V$, such that $s(u,x) \geq l \wedge u \rightsquigarrow x$. Let us call this event E_1.
2. If $\forall x \in V, \neg(s(x,v) \geq m \wedge x \rightsquigarrow v)$, then $(u \not\rightsquigarrow v)$ because $s(u,v) \geq m$. Therefore if $u \rightsquigarrow v$, then $\exists x \in V$, such that $s(x,v) \geq m \wedge x \rightsquigarrow v$. Let us call this event E_2.
3. $\exists c \in C_{u,v}^{l,m}$, $Cut(c)$ occurs, then $(u \not\rightsquigarrow v)$ by definition of cut. Therefore if $u \rightsquigarrow v$, $\forall c \in C_{u,v}^{l,m}$, $Cut(c)$ cannot occur. Let us call this event E_3.

Since E_1, E_2, E_3 are necessary conditions for the event $u \rightsquigarrow v$, therefore $R_{u,v} \leq Prob(E_1 \cap E_2 \cap E_3)$. $Prob(E_1) = out_u^l$ and $Prob(E_2) = in_v^m$. E_1 could only overlap with E_2 if there exists an edge (x,y) such that $s(u,x) \leq l-1 \wedge s(y,v) \leq m-1$. There will exist a path starting from u and *arriving* at v via the edge (x,y) with length $s(u,x)+1+s(y,v)$ which is less or equal than $l+m-1$. However, this is less than the shortest distance between u and v because $s(u,v) \geq l+m$ is given. This is a contradiction. Therefore E_1 cannot overlap with E_2. E_1 and E_2 also cannot overlap with E_3 because $\forall c \in C_{u,v}^{l,m}, \forall (x,y) \in c, s(u,y) > l \wedge s(x,v) > m$. Informally, any edges included in $C_{u,v}^{l,m}$, must be outside of the circle centred at u with radius l, and be outside of the circle centred at v with radius m. □

Given l and m, there are different choices of $C_{u,v}^{l,m}$. In this paper, we will simply define $C_{u,v}^{l,m}$ as $C_{u,v}^{l,m} = \{c_i | i \in \mathbb{I} \wedge l \leq i < s(u,v) - m\}$ where $c_i = \{(x,y) \mid s(u,x) = i \wedge s(u,y) = i+1 \wedge (x,y) \in E(g)\}$.

In the running example in Fig. 2 (b), the source and the destination is u and v respectively. The numbers on the edges are the existence probability. Suppose we have already indexed the outgoing/incoming probability up to the radius 2 and the shortest distance. The outgoing probability of radius 2 for u is the total proability of the possible worlds in which u can reach at least one of w_3 and w_4 whose shortest distance from u is 2. We initialize the upper bound as ${out^2}_u \times {in^2}_v$ and this upper bound can be further reduced by independent u-v cuts. In this case, $\{\{w_3w_5, w_4w_5\}\}$ is the only cut to be considered. Algorithm **Upper** formalizes the above statements.

Procedure. `Upper(`g, u, v`)`

```
1 begin
2   choose the l and m such that l + m < s(u,v) and out^l_u × in^m_v is minimum ;
3   upper = out^l_u × in^m_v ;
4   for i = l to s(u,v) − m − 1 do
5     c_i = {(x,y) | s(u,x) = i ∧ s(u,y) = i + 1 ∧ (x,y) ∈ E(g)} ;
6     upper = upper × (1 − ∏_{e∈c} p(e)) ;
7   return upper ;
8 end
```

5 Dynamic Monte Carlo Simulation

Traditionally, Monte Carlo simulation has been widely accepted as one of the efficient methods to answer reachability problems in uncertain graphs due to the NP-hard nature of this problem. In this section, we will propose a dynamic Monte Carlo simulation method which integrates sample generation and reachability test to maximize the computational sharing.

There are two major costs in Monte Carlo method:

1. **Generating Sample:** running a sample pool of size k requires to generate k samples, for each sample, every edge is to be assigned *exist* or *not exist* according to the existence probability. This costs $O(k|E|)$ of time.
2. **Checking Sample Reachability:** For each sample, we need to check the reachability between u and ve. This operation costs $O(|E|)$ of time for each iteration.

We have two observations in regarding to these two costs.

Observation 3: When some edges are missing, the presence of some other edges are no longer relevant. For example, in the example Fig. 2 (b), the states of other edges will no longer affect the reachability between u and v if uw_1 and uw_2 are missing.

Observation 4: Many samples share a significant portion of *existing* or *missing* edges, the reachability checking cost could be shared among them.

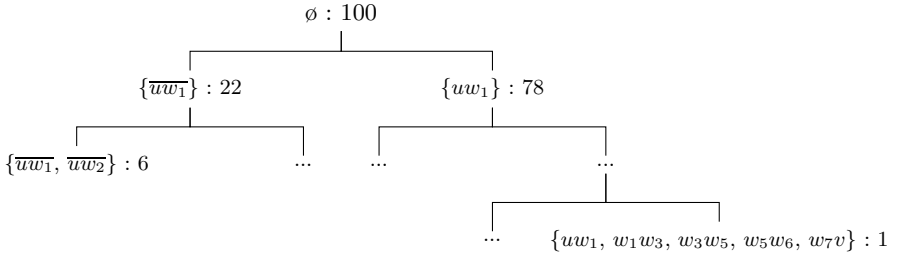


Fig. 3. A Dynamic Monte Carlo Sampling Example

Procedure. `DynamicMontecarlo(`g, s, t, p, k`)`

Input: g, u, v, t, k
1 $succ$, $fail = 0$;
2 $succ\_threshold = kt$;
3 $fail\_threshold = k(1-t)$;
4 $visited = \{s\}$;
5 $expand =$ outgoing edges of s ;
6 TestSample($visited$, $expand$, k, t) ;
7 **if** $succ \geq kp$ **then**
8 **return** *True*
9 **return** *False*

Procedure. `TestSample`(*visited, expand, n, t*)

Input: *visited*: visted vertices, *expand*: edges that can expand, n: number of samples in this group, v: destination

```
1  if expand = ∅ then
2      fail = fail + n ;
3      return
4  if u ∈ visited then
5      succ = succ + n ;
6      return
7  if succ ≥ succ_threshold Or fail > fail_threshold then
8      return
9  e = expand.pop_back ;
10 k1, k2 = 0 ;
11 visited2 = visited ⋃ v where v is the new vertex brought in by e;
12 expand2 = expand ⋃ v's outgoing edges which at least one end not in visited2;
13 foreach i = 0 to n do
14     r = random number from 0 to 1 ;
15     if r > p(e) then
16         k2 += 1 ;
17     else
18         k1 += 1 ;
19 TestSample(visited, expand, k1, t) ;
20 if succ ≥ kp Or fail > k(1 − p) then
21     return
22 TestSample(visited2, expand2, k2, t) ;
```

In our dynamic Monte Carlo method, starting with the source vertex u, we say u is already *reached*. An edge e is *expandable* if it starts from a *reached* vertex. We randomly pick an expandable edge e, then sample the existence of e for k iterations. The next step is to divide the samples into two groups, one group with e existing and another with e not existing. In the group with e existing, we can reach a new vertex w, and more edges become *expandable*. For both groups, we repeat the process of picking a random *expandable* edge, sampling its existence, and dividing the group into smaller batches. If a group contains no more *expandable* edges, the whole group cannot reach v. On the other hand, if v is contained in a group's *reached* vertices, then the whole group can reach v.

In the running example Fig. 2(b), we assume the number of samples to draw is 100. In the first step, we simultaneously poll the states of uw_1 for 100 samples, the result is shown in Fig. 3, uw_1 is *missing* in 22 of the 100 samples and *exists* in the rest. If uw_1 is failed, the next step is to poll on the other possible outgoing edge, uw_2, 6 of the 22 are failed in this case, and a u-v cut is formed in these 6 samples and the states of other edges are no longer relevent. If uw_1 is operational, we have more choices on which outgoing edges to poll next. If the destination can be reached in any step, we can conclude the corresponding sub-batch of samples are $u - v$ reachable. For example, the rightmost leaf node in Fig. 3.

Based on above observations, we present our Monte Carlo algorithm in Algorithm **DynamicMontecarlo** and Algorithm **TestSample**. In **DynamicMontecarlo**, we initializes a few global variables and invoke **TestSample** at line 6.

Then it checks whether the number of reachable samples is greater than kp. In **TestSample**, we firstly check whether there exist any more edges to expand. We could determine the whole group fails the reachability test if there is no more edges to expand. At line 9, we will randomly pick one edge to expand. From line 11 to line 12, we set up two groups which each represents the sample group in which the chosen edge is missing or is present, respectively. At line 13 to line 18, we will split all samples into these two group. At the end, it will recursively invoke TestSample. Line 7 to line 8, and line 20 to line 21 will check whether the current number of reachable or unreachable samples are enough to accept or reject the query.

Accuracy Guarantee: Let $R_{u,v}$ be the Probabilistic Reachability between u and v, the variance of the expected value, $E(R_{u,v})$ sampled by the Monte Carlo method is as following([16]:

$$\sigma^2(E(R_{u,v})) = \frac{R_{u,v} - {R_{u,v}}^2}{k} \tag{2}$$

As we introduce a threshold t into the Probabilistic Reachability Query, the result approximated by the Monte Carlo is correct as long as $R_{u,v} - E(R_{u,v}) \leq R_{u,v} - t$ when a query is rejected or $E(R_{u,v} - R_{u,v}) \leq t - R_{u,v}$.

Theorem 2 (Cantelli's Inequality [21]). *Suppose that r is a random variable with mean $E(r)$ and variance $\sigma^2(E(r))$, $Prob(r - E(r) \geq a) \leq \delta(a, \sigma(E(r))$ for any $a \geq 0$, where $Prob(r - E(r) \geq a)$ denotes the probability of $r - E(r) \geq a$, and $\delta(x, y)$ is defined as:*

$$\delta(x,y) = \begin{cases} 1 & \text{if } x = 1 \text{ and } y = 0 \\ 0 & \text{if } x = 0 \text{ and } y = 0 \\ \frac{1}{1+\frac{x^2}{y^2}} & \text{else} \end{cases}$$

Theorem 3. *The probability that the Monte Carlo method returns a false positive(or false negative) answer to a Probabilistic Reachability Query is less or equal than $\frac{{R_{u,v}-R_{u,v}}^2}{(k-1){R_{u,v}}^2-(2kt-1)R_{u,v}+kt^2}$, if we assume the exact probabilistic reachability values and the threshold follow uniform distribution.*

Proof. By using the Theorem 2, the probability that the Monte Carlo method returns a false negative answer:

$$\begin{aligned} Prob(R_{u,v} - E(R_{u,v}) \geq R_{u,v} - t) &\leq \delta(R_{u,v} - t, \sigma(E(R_{u,v})) \\ &= \frac{1}{1 + \frac{(R_{u,v}-t)^2}{\sigma(E(R_{u,v}))^2}} \\ &= \frac{{R_{u,v} - R_{u,v}}^2}{(k-1){R_{u,v}}^2 - (2kt-1)R_{u,v} + kt^2} \end{aligned}$$

Similarly, the probability that the Monte Carlo method returns a false positive answer can be deduced and the details are omitted. □

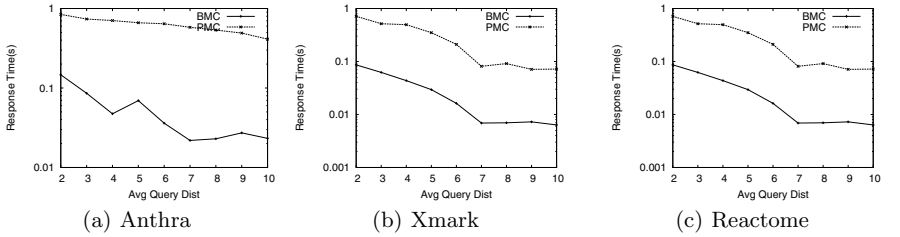
6 Experiment

We have performed extensive experiment to demonstrate our approach(Bounds and Dynamic Monte Carlo, or BMC) significantly outperforms plain Monte Carlo(PMC) simulation and Naive Enumeration. Note that, to the best of our knowledge, there are no other existing techniques aiming at efficiently support Probabilistic Reachabilityover large scale datasets. In the experiment, we used both real datasets and synthetic datasets to evaluate the performance. All experiments are conducted on a PC with 2.4GHz 4-core cpu, and 4GB main memory running Linux. Programs are implemented with C++. Every experiment is run against a group of 1000 randomly generated queries, and the average response time is taken as the result.

6.1 Real Dataset

In the experiments, we use 3 real datasets, Anthra, Xmark, and Reactome. All of these datasets were used by Jin in [14]. Anthra is a metabolic pathway from EcoCyc[1]. It contains 13736 vertices and 17307 edges. Xmark is a XML document containing 6483 vertices and 7654 edges. Reactome is a metabolic network with 3678 vertices and 14447 edges. We uniformly assign each edge a probability between 0 to 1. The index construction time for Anthra, Xmark, and Reactome is 11, 9, 16 seconds respectively. The index size is 1.5MB, 700KB, 120KB respectively. Please also note that the index size and construction time do not include the shortest path index since this depends on the technique chosen, which is not the scope of this paper.

As expected, Naive Enumeration cannot complete any experiment within 6 hours. This is because, out of the 1000 queries in each experiment, Naive Enumeration always freezes on at least one of them. The reason is that the cost of Naive Enumeration is almost the same as calculating the reachability if the probabilistic reachability between two vertices is close to the threshold. If the number of edges involved in the calculation is large, the enumeration cost is unaffordably expensive. This result shows that Naive Enumeration is not pratical in solving Probabilistic Reachability Queries. As a result, we will not include the experiment result for Naive Enumeration explicitly.

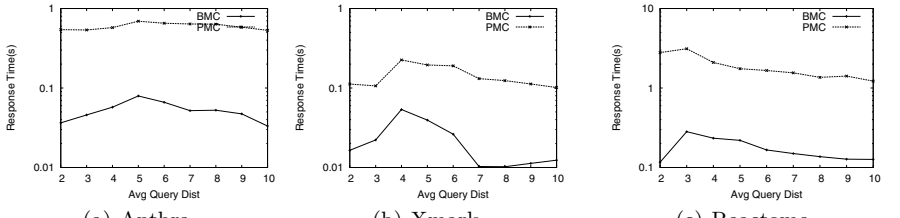


(a) Anthra (b) Xmark (c) Reactome

Fig. 4. Query Distance vs Response Time, $threshold = 0.6$

[1] www.ecocyc.org

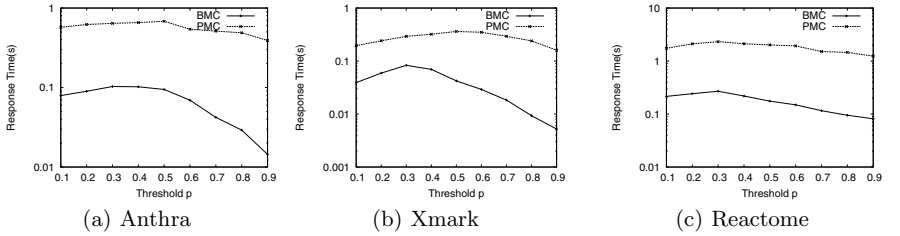
The first group of experiments studies how distance between query vertices affects the performance. The threshold is set at 0.6 and 1000 queries are issued. The result is shown in Fig. 4. The results show that BMC performs approximately one order of magnitude faster than PMC. An interesting observation is that as the distance increases, BMC and PMC both perform faster. In our analysis of this phenomenon, we have two observations: 1) It is generally much more time consuming to prove the destination is reachable than to prove it is not reachable in a possible world. Similarly, it is generally more time consuming to accept a query than to reject a query; 2) When the difference between the threshold and the probabilistic reachability is large, the Monte Carlo simulation requires less samples to answer the query, and also there is a better chance for the upper bound to be able to answer the query. These two observations can explain the above phenomenon. When the distance is small, generally speaking, the probabilistic reachability is higher, and thus the possibility for a state to be reachable is also higher. As the distance increases, the probabilistic reachability drops drastically. It means the average difference between probabilistic reachability and the threshold becomes larger. It also means the possibility for a state to be reachable becomes lower. We can also notice when the distance increases, the gap between BMC and PMC expands. This indicates the upper bound have played a more important role in this scenario.



(a) Anthra (b) Xmark (c) Reactome

Fig. 5. Query Distance vs Response Time, $threshold = 0.1$

In order to confirm our analysis, we repeat the experiment with a threshold of 0.1. The result is presented in Fig. 5. In this case, when the distance is small, BMC and PMC are both very efficient. This is because the average probabilistic reachability is much higher than 0.1. As the distance increases, the probabilistic reachability slowly drops towards the threshold and then keeps going lower. This represents the peak points of the three result graphs.

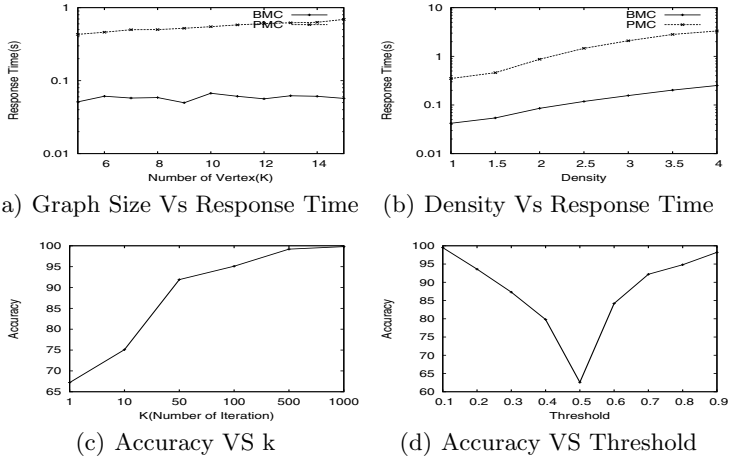


(a) Anthra (b) Xmark (c) Reactome

Fig. 6. Threshold vs Response Time, $QueryDistance = 0.6$

In next experiment, we fix the query distance to 5 and vary the threshold from 0.1 to 0.9. The result is shown in Fig. 6. We have observed that as the threshold increases, the response time generally increases slightly to a peak then drops drastically. This is because when the threshold is small, the probabilistic reachability is generally considerably higher than the threshold. As the threshold increases, it approaches the probabilistic reachability, thus the response time increases. In addition, as the threshold increases, more queries are rejected. To some extents, this effect offsets the increase caused by smaller gap between the threshold and the probabilistic reachability. This explains why the increase from 0.1 to the peak is moderate, as well as why the response time decreases drastically beyond the peak point.

The Anthra dataset has a similar density (average vertex degree) to Xmark but approximately 2 times the number of vertices. Reactome is much dense than both of Anthra and Xmark. As we can see from all of the above experiments, Anthra and Xmark have similar response time whereas Reactome is much slower than them. This suggests that the graph size will have limited effect on response time whereas the density plays a major role.



(a) Graph Size Vs Response Time (b) Density Vs Response Time

(c) Accuracy VS k (d) Accuracy VS Threshold

Fig. 7. Synthetic Data

6.2 Synthetic Dataset

In the first experiment, we generate graphs with 5000 to 15000 vertices with a fixed density of 1.5. The query distance is fixed at 5 and the threshold is set to 0.6. The result is shown in Fig. 7(a). We found that the increase of graph size has limited affect on BMC . This is because when the distance is fixed, the number of edges which can significantly affect the probabilistic reachability is somewhat unchanged. However, PMC's response time increases slightly as the graph size increases this is because PMC needs to draw a complete sample before testing the reachability.

In the second experiment, we generate graphs with density from 1 to 4 with a fixed graph size of 10000 vertices. The rest of the set up remains the same. The result is shown in Fig. 7(b). This shows the density will impact the response

time significantly. However, BMC 's response time increases 5 times from density of 1 to density of 4, whereas PMC's response time increases 10 times. This is because a portion of queries can be rejected by upper bounds. This portion of cost is affected less by the density.

6.3 Accuracy

Since Probabilistic Reachabilityis NP-hard, it is impractical to obtain probabilistic reachability precisely over large scale datasets. Thus, in this set of experiments, we use a small synthetic graph with 10 vertices to demonstrate the accuracy of the Monte Carlo simulation method. For the first experiment, we pick a pair of nodes whose reachability is approximately 0.7, and fix the threshold to be 0.85. We then vary the number of iteration k and the result is shown in Fig. 7 (c). We notice that the increase of k can dramatically increase the accuracy initially, and the increase diminishes when k grows larger. In the second part, we pick a pair of nodes whose reachability is approximately 0.5 and fix the number of iteration to be 100. We then vary the threshold from 0.1 to 0.9. The result is shown in Fig. 7 (d). In this case, the accuracy drops dramatically when the threshold approach 0.5 and again increases when the threshold moves further away from 0.5. This is because the Monte Carlo simulation will perform the worst when the threshold is very close to the probabilistic reachability.

7 Related Work

The Probabilitic Reachability problem has been studied in a number of papers from the 1970s on small scaled graphs, for example, [10], [13]. Valiant([10]) proved it is NP-hard in 1979. As Monte Carlo becomes the widely accepted method to approximate the answer, there are many studies([15], [16], [17], [18], [19]) to propose different sampling plans to reduce the estimation error.

There has been considerable effort put on the the certain reachability problem. A group of techniques([7], [5], etc) named chain decomposition, proposed to speed up online calculation of certain reachability by decomposing graphs into chains. Agrawal *et. al.* shows that using trees instead of chains is more efficient([1]). Based on the tree cover strategy, a few variants were proposed to improve Agrawal *et. al.*'s work. For example, Path-Tree([14]), Dual-Labeling([9]), Label+SSPI([2]), GRIPP([8]), etc. In [4], Cohen *et. al.* proposed a technique called 2-Hop. 2-Hop indexes each vertex with an in-set and an out-set which are used to infer the reachability between any two vertices. However, finding an optimal 2-Hop cover requires $O(n^4)$ time complexity. In order to improve the index building process, Cheng *et. al.* proposed an approximation 2-Hop cover([3]).

The techniques and applications of Uncertain Graphs have been studied in a number of recent papers, including mining frequent subpatterns([24], [25]), finding top-k maximal cliques([26]), etc.

8 Conclusion

In this paper, we study the problem of Probabilistic Reachability Queriesand proposed effective and efficient techniques to solve this problem. To the best of

our knowledge, we are the first to efficiently support Probabilistic Reachability Queries over large scale graphs using indexing techniques. We propose an index structure which assists in calculation of upper bound of probabilistic reachability efficiently. Should the bounds fail to answer a query, a dynamic Monte Carlo method is proposed to output an approximate answer. Through comprehensive experiments, we demonstrate that our solution is one order magnitude faster than the most widely accepted plain Monte Carlo simulation.

Acknowledgement

Xuemin Lin is supported by ARC Discovery Projects DP0881035, DP0987557, and DP110102937.

References

1. Agrawal, R., Borgida, A., Jagadish, H.V.: Efficient management of transitive relationships in large data and knowledge bases. In: SIGMOD, pp. 253–262 (1989)
2. Chen, L., Gupta, A., Kurul, M.E.: Stack-based algorithms for pattern matching on dags. In: VLDB, pp. 493–504 (2005)
3. Cheng, J., Yu, J.X., Lin, X., Wang, H., Yu, P.S.: Fast computation of reachability labeling for large graphs. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 961–979. Springer, Heidelberg (2006)
4. Cohen, E., Halperin, E., Kaplan, H., Zwick, U.: Reachability and distance queries via 2-hop labels. In: Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 937–946 (2002)
5. Jagadish, H.V.: A compression technique to materialize transitive closure. ACM Trans. Database Syst. 15(4), 558–598 (1990)
6. Schenkel, R., Theobald, A., Weikum, G.: HOPI: An efficient connection index for complex XML document collections. In: Hwang, J., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 237–255. Springer, Heidelberg (2004)
7. Simon, K.: An improved algorithm for transitive closure on acyclic digraphs. Theor. Comput. Sci. 58(1-3), 325–346 (1988)
8. Tribl, S., Leser, U.: Fast and practical indexing and querying of very large graphs. In: SIGMOD 2007: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 845–846 (2007)
9. Wang, H., He, H., Yang, J., Yu, P.S., Yu, J.X.: Dual labeling: Answering graph reachability queries in constant time. In: ICDE, p. 75 (2006)
10. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM J. Compt. 8, 410–421 (1979)
11. Jiang, B., Pei, J., Lin, X., Cheung, D.W., Han, J.: Mining preferences from superior and inferior examples. In: KDD, pp. 390–398 (2008)
12. Provan, J.S., Ball, M.O.: Computing Network Reliability in Time Polynomial in the Number of Cuts. Operations Research, Reliability and Maintainability 32(3), 516–526 (1984)
13. Shier, D.R., Liu, N.: Bounding the Reliability of Networks. The Journal of the Operational Research Society, Mathematical Programming in Honour of Ailsa Land 43(5), 539–548 (1992)
14. Jin, R., Xiang, Y., Ruan, N., Wang, H.: Efficiently Answering Reachability Queries on Very Large Directed Graphs. In: SIGMOD (2008)

15. Easton, M.C., Wong, C.K.: Sequential Destruction Method for Monte Carlo Evaluation of System Reliability. IEEE, Reliability 29, 191–209 (1980)
16. Fishman, G.S.: A Monte Carlo Sampling Plan for Estimating Network Reliability. Operational Research 34(4), 581–594 (1986)
17. Karp, R., Luby, M.G.: A New Monte Carlo Method for Estimating the Failure Probability of An N-component System. In: Computer Science Division. University of California, Berkley (1983)
18. Okamoto, M.: Some Inequalities Relating To the Partial Sum of Binomial Probabilities. Annals Inst. Statistical Mathematics 10, 29–35 (1958)
19. Fishman, G.S.: A Comparison of Four Monte Carlo Methods for Estimating the Probability of s-t Connectedness. IEEE, Trans. Reliability 35(2) (1986)
20. Chan, E.P., Lim, H.: Optimization and Evaluation of Shortest Path Queries. VLDB Journal 16(3), 343–369 (2007)
21. Meester, R.: A Natural Introduction to Probability Theory (2004)
22. Adar, E., Ré, C.: Managing Uncertainty in Social Networks. Data Engineering Bulletin 30(2), 23–31 (2007)
23. Zou, Z., Gao, H., Li, J.: Discovering Frequent Subgraphs over Uncertain Graph Databases under Probablistic Semantics. In: KDD (2010)
24. Zou, Z., Li, J., Gao, H., Zhang, S.: Mining Frequent Subgraph Patterns from Uncertain Graph Data. TKDE 22(9), 1203–1218 (2010)
25. Zou, Z., Gao, H., Li, J.: Discovering Frequent Subgraphs over Uncertain Graph Databases under Probabilistic Semantics. In: SIGKDD, pp. 633–642 (2010)
26. Zou, Z., Li, J., Gao, H., Zhang, S.: Finding Top-k Maximal Cliques in an Uncertain Graph. In: ICDE, pp. 649–652 (2010)