Efficient Subgraph Search over Large Uncertain Graphs

Ye Yuan[§] Guoren Wang[§♯] Haixun Wang[†] Lei Chen[‡]
[§]College of Information Science and Engineering, Northeastern University, China
[†]Microsoft Research Asia
[‡]Hong Kong University of Science and Technology, Hong Kong, China
[♯]State Key Lab of Software Engineering Wuhan University, Wuhan, Hubei, China
{yuanye,wanggr}@ise.neu.edu.cn, haixunw@microsoft.com, leichen@cse.ust.hk

ABSTRACT

Retrieving graphs containing a query graph from a large graph database is a key task in many graph-based applications, including chemical compounds discovery, protein complex prediction, and structural pattern recognition. However, graph data handled by these applications is often noisy, incomplete, and inaccurate because of the way the data is produced. In this paper,we study subgraph queries over uncertain graphs. Specifically, we consider the problem of answering threshold-based probabilistic queries over a large uncertain graph database with the possible world semantics. We prove that problem is #P-complete, therefore, we adopt a *filtering-and-verification* strategy to speed up the search. In the *filtering* phase, we use a probabilistic inverted index, PIndex, based on subgraph features obtained by an optimal feature selection process. During the *verification* phase, we develop exact and bound algorithms to validate the remaining candidates. Extensive experimental results demonstrate the effectiveness of the proposed algorithms.

1. INTRODUCTION

In this paper, we study the problem of subgraph matching over large *uncertain* graphs. A large variety of applications work on graph structured data, and in many cases, the graph data they deal with are uncertain or noisy by nature [1, 4, 5, 8, 14, 24, 21, 20, 31].

For example, in bioinformatics, protein-protein interaction (PPI) networks obtained through experiments are noisy – they may contain interactions that do not really exist and at the same time they may miss real interactions [8, 26, 31]. It is thus more natural to represent a PPI network as an uncertain graph where nodes (proteins) are connected by uncertain edges associated with numerical values which indicate the possibility of interaction between the proteins. As another example, in visual pattern recognition, graphs are used to model visual objects, and since information is incomplete or noisy, such representations are uncertain [4, 21, 24]. It has been shown that methods finding probabilistic matches outperform exact matching algorithms [34] in many aspects. Uncertainty also arises in social networks: links between two persons are often associated with probabilities that represent the uncertainty of the link [20] or the strength of influence a person has over another person in virtual marketing [12].

Proceedings of the VLDB Endowment, Vol. 4, No. 11

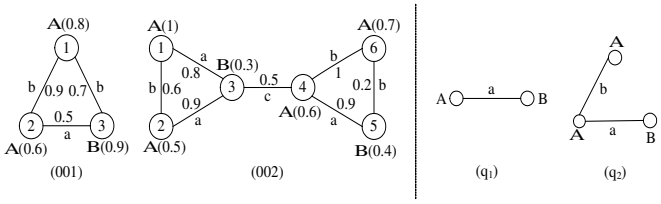


Figure 1: Uncertain graph database & Query graphs.

In the aforementioned applications, graph matching is a typical query for many interesting tasks, such as identifying scenes (graphs) in visual pattern recognition [4, 21], predicting complex biological interactions (graphs) [8, 31], and finding social communities (graphs) [12]. Therefore, it is important to study subgraph matching over large uncertain graphs.

1.1 Probabilistic Subgraph Matching

In this paper, we focus on *threshold-based probabilistic subgraph matching* (T-PS) over a large set of uncertain graphs. Specifically, let $D = \{g_1, g_2, ..., g_n\}$ be a set of uncertain graphs, let q be a query graph, and let ϵ be a probability threshold, a T-PS query retrieves all graphs $g \in D$ such that the *subgraph isomorphic probability* (SIP) between q and g is not smaller than ϵ. We will formally define SIP in Section 2.

Example 1. *Figure 1.1 shows a database that contains two uncertain graphs (001 and 002) and two query graphs (q_1 and q_2). Vertices and edges are labeled (A, B, C, ...; a, b, c, ...), and a real number associated with each vertex and each edge represents the existence probability of the vertex or edge.*

The first question we must answer is, what constitutes a match in uncertain graphs? To answer this question, we employ the *possible world semantics* [30, 11], which has been used for modeling query processing over probabilistic databases. A possible world graph (PWG) of is a possible *instance* of an uncertain graph. It contains a subset of vertices and edges of the uncertain graph, and it has a weight which is the product of the probabilities of all the vertices and edges it has. Then, for a query graph q and an uncertain graph g, the probability that q matches g is the summation of the weights of those PWGs of g that are subgraph-isomorphic to q.

Example 2. *Figure 1.1 lists all the PWGs of uncertain graph 001 and their weights. Altogether there are 18 PWGs for graph 001, and the sum of all the weights is 1. To decide if q_1 matches uncertain graph 001, we first find all of 001's PWGs that contain q_1 as a subgraph. Note that, "g contains q as a subgraph" means*

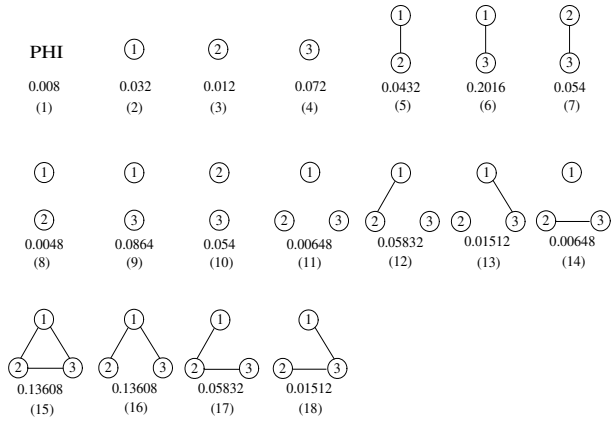


Figure 2: Possible world graphs of uncertain graph 001.

that q is subgraph-isomorphic to g. The results are PWGs 7, 14, 15, 17 and 18. Next, we add up the probabilities of these PWGs: $0.054 + 0.00648 + 0.13608 + 0.05832 + 0.01512 = 0.27072$*. If, for example, the query specifies a threshold of 0.4, then q_1 fails to match uncertain graph 001 since* $0.27072 < 0.4$.

The above example gives a naive solution to T-PS query processing. We call it $SCAN$, as it needs to enumerate all PWGs of an uncertain graph, and conducts the subgraph isomorphism test for each PWG. $SCAN$ is very inefficient due to the exponential number of PWGs and the hard problem of subgraph isomorphism test. In fact, it is not difficult to see that $SCAN$ is NP-complete [13]. Clearly, an efficient solution is preferred. In this paper, we propose a *filtering-and-verification* method to solve this problem.

1.2 Our Method and Our Contributions

For a dataset of uncertain graphs $D = \{g_1, ..., g_n\}$ and a query graph q, we perform T-PS query in three steps, namely, structural pruning, probabilistic pruning, and verification.

Structural Pruning

The idea of structural pruning is straightforward. If we remove the uncertainty[1] in an uncertain graph, and the resulting graph still does not contain q, then the original uncertain graph cannot contain q. Formally, for $g \in D$, let g^c denote the *corresponding deterministic graph* after removing the uncertainty information from g. We have

Theorem 1. *If $q \nsubseteq g^c$, $Pr(q \subseteq g) = 0$.*

where $\subseteq$ denotes graph isomorphic relationship, and $Pr(q \subseteq g)$ denotes the subgraph isomorphism probability for q in g.

Based on this observation, given D and q, we can prune the database $D^c = \{g_1^c, ..., g_n^c\}$ using conventional exact graph matching methods. In this paper, we adopt the method in [29] to scan the index once to compute the final results. Though there are many works [29, 33, 7, 16, 15] that can be used for structural pruning, theoretical and evaluation results in [7] show that [29] has more efficient runtime than other works. Assume the result after structural pruning is $SC_q^c = \{g^c | q \subseteq g^c, g^c \in D^c\}$. Then, its corresponding uncertain graph set, $SC_q = \{g | g^c \in SC_q^c\}$, is the input for uncertain graph matching in the next step. Note that we need to build a tree index [29] for all deterministic graphs. In query processing, we only scan the tree index once to compute final results, which is quite efficient.

[1]That is, set the probability of each uncertain edge and vertex to 1.

Probabilistic Pruning

To further prune the results, we propose a *probabilistic inverted index* (PIndex) for probabilistic pruning. For a given set of uncertain graphs D and its corresponding set of deterministic graphs D^c, we create a feature set F from D^c, where each feature is a deterministic graph, i.e., $F \subset D^c$. For each $f \in F$, we create an inverted index D_f :

$$D_f = \{\langle g, Pr(f \subseteq g)\rangle | f \subseteq g^c, g \in D\}$$

where $Pr(f \subseteq g)$ is the subgraph isomorphism probability. A challenging issue in this step is selecting features for PIndex. As we will show in Section 3.2, any frequent subgraphs may have good pruning power. However, it would be unrealistic to index all of them. One of our contributions in this work is cost model-based feature selection. We start with frequent subgraphs denoted as F_0 and devise a model to estimate query cost. Then, we select the best feature set $F \subset F_0$ that optimizes the cost model. We prove that solving the exact optimization problem is hard and we propose an approximate approach, which enables us to derive an inverted index of small size and powerful pruning capability.

To avoid computing $Pr(f \subseteq g)$ directly (#P-complete), we approach $Pr(f \subseteq g)$ with its lower and upper bounds: $LowerB(f)$ and $UpperB(f)$. Both $LowerB(f)$ and $UpperB(f)$ can be calculated efficiently. Furthermore, we convert the problem of computing bounds into the maximum clique problem so that we can obtain tight bounds. Most important of all, we derive two pruning conditions to filter out uncertain graphs for a query q as follows.

Pruning 1. Given a query q and a threshold ϵ, if $\exists f \in F$ such that $q \supseteq f$ and $Pr(f \subseteq g) < \epsilon$, then g can be safely pruned.

Pruning 2. Given a query q and a threshold ϵ, if $\exists f \in F$ such that $q \subseteq f$ and $Pr(f \subseteq g) \geq \epsilon$, then g is in the final answers.

Verification

In this step, we calculate $Pr(q \subseteq g)$ for query q and candidate answer g to make sure g is really an answer, i.e. $Pr(q \subseteq g) \geq \epsilon$. Our contribution in this step is that we utilize a very tight bound on the output of each step during the computation of the exact SIP, which enables us to stop the computation as early as possible.

1.3 Paper Organization

The rest of this paper is organized as follows. We formally define T-PS queries over uncertain graphs in Section 2. Section 3 presents algorithms for probabilistic pruning, where an optimal feature selection strategy is used to maximize probabilistic pruning. We propose exact and bound algorithms for calculating SIP in Section 4. We discuss the results of performance tests on real data sets in Section 5 and the related work in Section 6. We conclude our work in Section 7.

2. PROBLEM DEFINITION

In this section, we define some necessary concepts and show the complexity of our problem. Table A in Appendix summarizes the notations used in this paper.

Following the proposed uncertain graph models in pervious works [17, 25, 35, 36], we define the uncertain graph as follows.

Definition 1. ***(Uncertain Graph)*** *Let $g^c = (V, E, \Sigma, L)$ be an undirected deterministic graph[2] where V is a set of vertices, E is a set of edges, Σ is a set of labels, and $L : V \cup E \rightarrow \Sigma$ is a function that assigns labels to vertices and edges. An uncertain*

[2]In this paper, we consider undirected graphs, although it is straightforward to extend our methods to other types of graphs.

graph is defined as $g = (g^c, P_V, P_E)$, where $P_V : V \to [0, 1]$ is a function that assigns existence probabilities to vertices in V , and $P_E : E \to [0, 1]$ is a function that assigns existence probabilities[3] *to edges in E.*

This model is *complete*, since it can represent any pdf of a probabilistic graph. We denote g^c as g's corresponding deterministic graph. Clearly, g^c is a special uncertain graph with $P_V = 1$ and $P_E = 1$. We refer the size of g^c or g as the size of its edge set.

Definition 2. ***(Possible World Graph)*** *A possible world graph $g' = (V', E', \Sigma', L')$ is an instantiation of an uncertain graph $g = ((V, E, \Sigma, L), P_V, P_E)$, where $V' \subseteq V$, $E' \subseteq E \cap (V' \times V')$, $\Sigma' \subseteq \Sigma$, and L' is the function obtained by restricting L to $V' \cup E'$. We denote the relationship between g' and g as $g \Rightarrow g'$.*

Both g' and g^c are deterministic graphs. But an uncertain graph g corresponds to one g^c and multiple possible world graphs. We use $PWG(g)$ to denote the set of all possible world graphs derived from g. For example, Figure 1.1 listed all the 18 possible world graphs of the uncertain graph 001 in Figure 1.1.

Following the convention in [4, 8, 24, 21, 23, 26, 31, 34], we assume the existences of different edges or different vertices in an uncertain graph are independent. Then, the probability of a possible world graph g' is given by [35, 36]:

$$Pr(g \Rightarrow g') = \prod_{v \in V'} P_V(v) \prod_{v \in V \setminus V'} (1 - P_V(v)) \prod_{e=(u,v) \in E'} P_E(e|u, v) \prod_{e=(u,v) \in E \cap (V' \times V') \setminus E'} (1 - P_E(e|u, v)). \quad (1)$$

Clearly, for any possible world graph g', we have $Pr(g \Rightarrow g') > 0$ and $\sum_{g' \in PWG(g)} Pr(g \Rightarrow g') = 1$, that is, each possible world graph has an existence probability, and the sum of these probabilities is 1.

We are not the first to assume the independence of vertices or edges in an uncertain graph [17, 35, 36, 25]. Furthermore, we make the assumption based on many real applications [31, 8, 26, 24, 21, 4, 34, 23]. For example, in PPI networks, biologists first establish elementary links with probabilities between proteins, then use machine learning tools to predict other possible links based on the elementary links. The elementary links are assumed to be independent of each other, and models are trained over the marginal probabilities. Their empirical studies show the assumption works well and the result is consistent with the ground truth[4]. As another example, in uncertain social networks [1] modeled by a probabilistic relational database, nodes are assumed to be independent as well.

Definition 3. ***(Subgraph Isomorphism)*** *Given two graphs $g_1 = (V_1, E_1, \Sigma_1, L_1)$ and $g_2 = (V_2, E_2, \Sigma_2, L_2)$, we say g_1 is subgraph isomorphic to g_2 (denoted by $g_1 \subseteq g_2$), if and only if there is an injective function $f : V_1 \to V_2$ such that:*

- *for any $(u, v) \in E_1$, $(f(u), f(v)) \in E_2$;*
- *for any $u \in V_1$, $L_1(u) = L_2(f(u))$;*
- *for any $(u, v) \in E_1$, $L_1(u, v) = L_2(f(u), f(v))$.*

The subgraph (V_3, E_3) of g_2 with $V_3 = \{f(v)|v \in V_1\}$ and $E_3 = \{(f(u), f(v))|(u, v) \in E_1\}$ is called the embedding of g_1 in g_2.

When g_1 is subgraph isomorphic to g_2, we also say that g_1 is a subgraph of g_2 and g_2 is a super-graph of g_1.

[3]These are conditional probabilities given that endpoints of the edges exist.

[4]More details can be found in http://string-db.org.

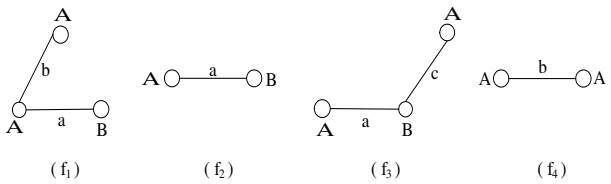


Figure 3: Features of uncertain graph database.

Definition 4. ***(Subgraph Isomorphism Probability)*** *For a given query graph q and an uncertain graph g, we define their subgraph isomorphism probability as*

$$Pr(q \subseteq g) = \sum_{g' \in SI(q,g)} Pr(g \Rightarrow g') \quad (2)$$

where $SI(q, g)$ is g's possible world graphs that are subgraph isomorphic to q, that is, $SI(q, g) = \{g' \in PWG(g)|q \subseteq g'\}$.

Definition 5. ***(Probabilistic Subgraph Query)*** *Given a set of uncertain graphs $D = \{g_1, ..., g_n\}$, a query graph q, and a probability threshold ϵ ($0 < \epsilon \leq 1$), a subgraph query returns a set of uncertain graphs $\{g \in D|Pr(q \subseteq g) \geq \varepsilon\}$.*

From Definition 5, we know that in order to answer probabilistic subgraph queries efficiently, we must be able to calculate SIP (subgraph isomorphism probability) efficiently. We now give the complexity of calculating SIP.

Theorem 2. *It is #P-complete to calculate the subgraph isomorphism probability.*

Proof. The proof is included in Appendix A.1. ■

3. PROBABILISTIC PRUNING

As we mentioned in Section 1.2, we first conduct structural pruning to remove uncertain graphs that do not contain the query graph g. In this step, we remove the uncertainty in an uncertain graph and adopt the method proposed in [29] for pruning the deterministic graphs. Thus, the focus of this paper lies on the second and the third steps: probabilistic pruning and verification. We discuss probabilistic pruning and verification techniques in detail in this and the next sections, respectively.

3.1 Upper and Lower Bounds

We introduced two probabilistic pruning conditions in Section 1.2. However, to use these two conditions, we have to compute $Pr(f \subseteq g)$, which is proven to be #P-complete. Therefore, we approach $Pr(f \subseteq g)$ using its upper and lower bounds and conduct pruning using the two bounds.

3.1.1 Pruning Conditions

Assuming we already have the lower bound $LowerB(f)$ and the upper bound $UpperB(f)$ of $Pr(f \subseteq g)$, we can rewrite the pruning conditions given in Section 1.2 as follows.

Pruning 1. Given a query q, a threshold ϵ, if $\exists f \in F$ such that $q \supseteq f$ and $UpperB(f) < \epsilon$, then g can be safely pruned from SC_q.

Pruning 2. Given a query q and a threshold ϵ, if $\exists f \in F$ such that $q \subseteq f$ and $LowerB(f) \geq \epsilon$, then g is the final answers, i.e., $g \in A_q$, where A_q is the final answer set.

Before proving the correctness of the above two pruning conditions, we first introduce a lemma about $Pr(q \subseteq g)$, which will be used for the proof. Let $Eq = \{q_1, .., q_{|Eq|}\}$ be the set of all embeddings of q in the deterministic graph g^c, Bq_j be a boolean variable for $1 \leq j \leq |Eq|$, and $Pr(Bq_j)$ be the probability that the embedding q_j exists in g. We have

Lemma 1.

$$Pr(q \subseteq g) = Pr(Bq_1 \vee ... \vee Bq_{|Eq|}). \quad (3)$$

Proof. The proof is included in Appendix A.2. ■

With Lemma 1, we can prove the two pruning conditions.

Theorem 3. *Given a query q and threshold ϵ, if $\exists f \in F$ such that $q \supseteq f$ and $UpperB(f) < \epsilon$, then g can be safely pruned from SC_q.*

Proof. The proof is included in Appendix A.3. ■

Theorem 4. *Given a query q and a threshold ϵ, if $\exists f \in F$ such that $q \subseteq f$ and $LowerB(f) \geq \epsilon$, then g is in the final answers, i.e., $g \in A_q$.*

Proof. The proof is included in Appendix A.4. ■

Note that the pruning process needs to address the traditional subgraph isomorphism problem ($q \subseteq f$ or $q \supseteq f$). In our work, we implement the state-of-the-art method VF2 [9] that gives a fast solution to subgraph isomorphism.

3.1.2 Deriving the Upper and Lower Bounds

In this subsection, we discuss how to efficiently compute the two bounds $LowerB(f)$ and $UpperB(f)$.

From Lemma 1, we have:

$$\begin{aligned} Pr(f \subseteq g) &= Pr(Bf_1 \vee ... \vee Bf_{|Ef|}) \\ &= 1 - Pr(\overline{Bf_1} \wedge ... \wedge \overline{Bf_{|Ef|}}). \end{aligned} \quad (4)$$

Boolean variables Bf_i, $1 \leq i \leq |Ef|$, may have common parts, thus $\overline{Bf_i}$, $1 \leq i \leq |Ef|$, are not independent, which means $Pr(\overline{Bf_1} \wedge ... \wedge \overline{Bf_{|Ef|}}) \neq \prod_{i=1}^{|Ef|} Pr(\overline{Bf_i})$. If we choose a group of features, among $|Ef|$ boolean variables, that cannot have common parts, their corresponding boolean variables are independent each other. For example, consider uncertain graph 002 and feature f_1 in Figures 1.1 and 3. Figure 3.1.2 shows 3 embeddings, of f_1 in 002, in which $EM3$ does not have common parts with $EM1$ or $EM2$. Thus the boolean variable of $EM3$ is independent with the boolean variable of $EM1$ or $EM2$.

Let Bf_j, $1 \leq j \leq IN \leq |Ef|$, be IN independent boolean variables. Since $\vee_{j=1}^{IN} Bf_j \subseteq \vee_{i=1}^{|Ef|} Bf_i$, then we can obtain a lower bound of $Pr(f \subseteq g)$ as follows,

$$\begin{aligned} Pr(f \subseteq g) &= Pr(\vee_{i=1}^{|Ef|} Bf_i) \geq Pr(\vee_{j=1}^{IN} Bf_j) \\ &= 1 - \prod_{j=1}^{IN} (1 - Pr(Bf_j)) \\ &= LowerB(f). \end{aligned} \quad (5)$$

Equation 5 gives a lower bound of $Pr(f \subseteq g)$. As shown in Theorem 2, it is hard to calculate $Pr(f \subseteq g)$. However we need to compute the $LowerB(f)$ efficiently, which is important to construct PIndex efficiently. Fortunately, as shown in Equation 5, if we can compute $Pr(Bf_i)$ efficiently, we can easily get the lower bound. Therefore, the following theorem states how to compute $Pr(Bf_i)$.

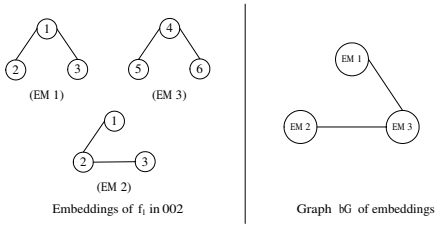


Figure 4: Embeddings & bG of feature f_1 in uncertain graph 002.

Theorem 5.

$$Pr(Bf_i) = \prod_{v \in f_i} P_V(v) \prod_{e=(u,v) \in f_i} P_E(e|u, v) \quad (6)$$

Proof. The proof is included in Appendix A.5. ■

Obtain Tightest Lower Bound. Among all $|Ef|$ boolean variables, there are many groups which contain independent boolean variables, which leads to different lower bounds. However, we want to get a tight lower bound in order to increase the pruning power. We can solve the problem as follows:

We construct an undirected graph, bG, with each *node* representing a boolean variable Bf_i, $1 \leq i \leq |Ef|$, and a *link* connecting two independent boolean variables (nodes). Note that, in this paper, to avoid confusions, *nodes* and *links* are used for bG, while *vertices* and *edges* are for uncertain graphs. In bG, a *clique* is a set of nodes such that any two nodes of the set are adjacent. The *size* of a clique is the number of nodes it contains. A *maximum clique* is a clique that has the largest size among all cliques. According to Equation 5, we assign each node a weight, $-\ln(1 - Pr(Bf_i))$, and define the weight of a clique as the sum of node weights in the clique. Clearly, given a clique in bG with weight v, the lower bound of $Pr(f \subseteq g)$ is $1 - e^{-v}$. Thus, the larger the weight, the tighter (larger) the lower bound. To obtain a tight lower bound, we should find a clique whose weight is the largest, which is exactly the *maximum weight clique* problem. It is known that the maximum weight clique problem is NP-hard [13]. In [3], the maximum weight clique problem is formulated as a 0-1 integer programming. The integer programming is relaxed to be a linear programming whose dual linear program, called *weighted fractional minimum node coloring* problem. The dual linear program returns a very tight upper bound, z, on the weight of maximum weight clique and there is an efficient algorithm that can solve the weighted fractional minimum node coloring problem [3]. Therefore, we use $1 - e^{-z}$ computed by the efficient solution [3] as the lower bound for $Pr(f \subseteq g)$.

Example 3. *Consider an uncertain graph 002 and a feature f_1 in Figures 1.1 and 3. Figure 3.1.2 shows 3 embeddings, of f_1 in 002, in which $EM3$ does not have common parts with $EM1$ or $EM2$. Based on the above discussion, we construct bG, for the 3 embeddings, shown in Figure 3.1.2. There are two maximum cliques namely, $EM1$ and $EM3$, $EM2$ and $EM3$. According to Equation 5, the lower bounds derived from the 2 maximum cliques are 0.272 and 0.275 respectively. Therefore we select the larger (tighter) value 0.275 to be the lower bound of f_1 in 002.*

Recall that $Pr(f \subseteq g) \neq 1 - \prod_{i=1}^{|Ef|}(1 - Pr(Bf_i))$ due to non-independent boolean variables Bf_i for $1 \leq i \leq |Ef|$. However, the boolean variables satisfy inequality:

$$Pr(\overline{Bf_1} \wedge ... \wedge \overline{Bf_{|Ef|}}) \geq \prod_{i=1}^{|Ef|} Pr(\overline{Bf_i})$$

Then, Equation 4 can be written as

$$\begin{aligned} Pr(f \subseteq g) &\leq 1 - \prod_{i=1}^{|Ef|} Pr(\overline{Bf_i}) \\ &= 1 - \prod_{i=1}^{|Ef|} (1 - Pr(Bf_i)) \\ &= UpperB(f). \end{aligned} \tag{7}$$

Equation 7 gives an upper bound of $Pr(f \subseteq g)$.

Please refer to Appendix A.7 for values and calculation time of upper and lower bounds of SIP.

Note that the above techniques need to compute embeddings of q or f in g^c. In this paper, we implement the efficient algorithm in [32] to compute embeddings of a query in a deterministic graph.

3.2 Optimal Feature Selection for Probabilistic Inverted Index

As introduced in the last subsection, there are two conditions, i.e., $CND \triangleq (q \supseteq f \wedge UpperB(f) < \epsilon) \vee (q \subseteq f \wedge LowerB(f) \geq \epsilon)$, considered in the probabilistic pruning phase.

According to the conditions, it is not difficult to see that any frequent feature, no matter its size, should be indexed to maximize the pruning power of probabilistic inverted index (PIndex). However, there can exist thousands or millions of features, and it would be unrealistic to index all of them. Thus, our goal is to maximize the pruning capability of PIndex with a small number of indexed features. Motivated by *machine learning* methods for query processing [28, 6], in this section, we employ a model, which uses a query log Γ as the training data, to select features offline. Based on the model, we develop an optimal selection mechanism to remove useless features so that PIndex can have a great pruning capability.

For an initial set of features $F_0 = \{f_1, f_2, ..., f_m\}$, we would like to select a subset $F \subset F_0$ to maximize the pruning capability. Frequent subgraph mining algorithms, i.e., [32] can be used to generate the initial feature set. Next, we employ an optimization model to select the feature set F using a query log set Γ as the training data.

Let C_q be the candidate set after probabilistic pruning.

Recall that the naive solution, $SCAN$, to the T-PS search problem examines the uncertain database SC_q sequentially and computes SIP for each uncertain graph to decide whether its SIP is not smaller than ϵ. Given a single query q, we first do the structural pruning and get corresponding SC_q. Then, we define the *gain*, J, of indexing a feature set F as the number of SIP computations that can be saved from $SCAN$:

$$\begin{aligned} J &= |SC_q| - |C_q| - |F| \\ &= |\cup_{q \doteq CND} \{g | g \in SC_q\}| - |F| \end{aligned} \tag{8}$$

where $CND \triangleq (q \supseteq f \wedge UpperB(f) < \epsilon) \vee (q \subseteq f \wedge LowerB(f) \geq \epsilon)$.

To obtain more effective features, we use a set of queries $\{q_1, q_2, ..., q_w\}$ instead of a single query in Γ. In this case, an optimal index should maximize the total gain

$$J_{total} = \sum_{l=1}^{w} |\cup_{q_l \doteq CND} \{g | g \in SC_{q_l}\}| - w|F| \tag{9}$$

which is a summation of the gain in Equation 8 over all queries.

According to Equation 9, we should index a feature as long as q satisfies CND, and f covers at least one uncertain graph which has not yet been covered by other features (a feature covers an uncertain graph g if g^c contains it).

	001	002
f_1	(0.19,0.19)	(0.27,0.49)
f_2	(0.27,0.27)	(0.4,0.49)
f_3	0	(0.01,0.11)

Figure 5: Feature-graph matrix.

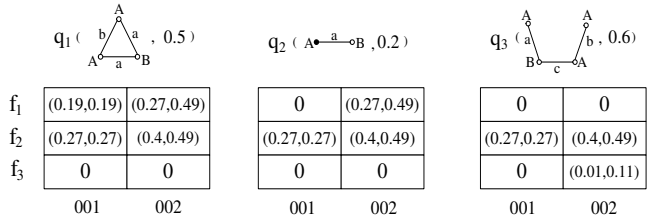


Figure 6: Probabilistic matrix.

Note that Chen et al. propose a similar cost model [6] to generate subgraph features for deterministic super-graph search. However, this model focuses on the pruning condition for the deterministic super-graph query, while our model considers probabilistic pruning rules (i.e. CND) for uncertain subgraph search. Thus its solution cannot be applied to the proposed model in this paper. In the sequel, we gives an efficient algorithm to select features based on Equation 9.

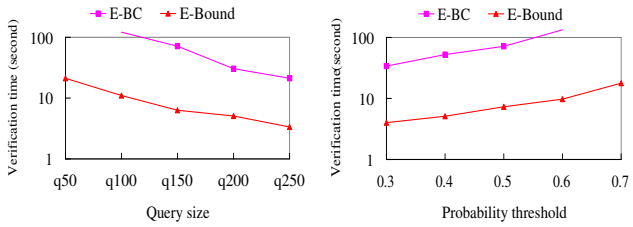
We first build a *feature-graph matrix* to display the relationship between features and uncertain graphs, whose (i, j)-entry gives $\{LowerB(f), UpperB(f)\}$ of the ith feature in jth uncertain graph if the uncertain graph's corresponding deterministic graph contains that feature, and otherwise gives value 0. Then, we introduce the concept of *probabilistic graph matrix*, which is derived from the feature-graph matrix but with its (i, j)-entry set to value 0 if the query does not satisfy CND.

Example 4. *For uncertain graphs in Figure 1.1, the feature-graph matrix of f_1, f_2 and f_3, in Figure 3, is given in Figure 3.2. Figure 3.2 shows three query graphs and their corresponding probabilistic graph matrix for the 3 features and 2 uncertain graphs.*

Recall that in query log, we record total w queries to optimize Equation 9. For w queries, we build a probabilistic graph matrix **M** with row size equaling to the number of initial feature set F_0 and column size equaling to $w \times |SC_q|$. For example, in Figure 3.2, features $\{f_1, f_2, f_3\}$ compose of the initial feature set. Entries of **M** give corresponding $\{LowerB(f), UpperB(f)\}$ in Figure 3.2 if uncertain graphs 001 and 002 satisfy 3 queries, otherwise give values 0. With such a probabilistic graph matrix, we can map our cost model to a well-known NP-complete problem [13], defined as follows:

Definition 6. ***(Maximum Coverage**, [6]) Given a set of subsets $\boldsymbol{S} = \{S_1, ..., S_m\}$ of the universal set $U = \{1, 2, ..., n\}$ and a weight η associated with any $S_i \in \boldsymbol{S}$, find a subset $\mathbf{Z}$ of $\boldsymbol{S}$ such that $|\cup_{S_i \in \mathbf{Z}} S_i| - \eta|\mathbf{Z}|$ is maximized.*

For a probabilistic graph matrix **M**, U corresponds to the column index of **M**, **S** corresponds to non-zero entries in each row of **M**, **Z** corresponds to the selected rows, and η corresponds to the number of queries w. Take the matrix in Figure 3.2 as an example: $U = \{1, 2, 3, 4, 5, 6\}$ because there are 6 columns in the concatenated matrix, $S = \{\{1, 2, 4\}, \{1, 2, 3, 4, 5, 6\}, \{6\}\}$, i.e., $\{1, 2, 4\}$



(a) (b)

Figure 7: Verification time on real dataset.

corresponds to the first row in the matrix $\eta = 3$ since there are 3 queries.

Since the maximum coverage problem is NP-complete, we follow its greedy algorithm [13] to approximate the optimal features. In each iteration, we select a row with the most number of non-zero entries from the matrix **M**. Given a probabilistic graph matrix **M**, the algorithm first selects a row that contains most non-zero entries, and then removes the columns covered by these non-zero entries. This process repeats until all columns are covered by the removed rows. The greedy algorithm can approximate the optimal index within a ratio of $1 - 1/e$ [13].

The introduced PIndex algorithm builds a flat index structure, where each feature is tested sequentially against any input query. Clearly, such a flat index is inefficient to answer a query. To avoid the sequence test, we construct a *prefix-tree structured index* for PIndex. We give the details about this index in Appendix A.6.

4. VERIFICATION

In this section, we present the algorithms to compute subgraph isomorphism probability (SIP) of an uncertain graph g in the final answers, i.e., $A_q = \{g \in C_q | Pr(q \subseteq g) \geq \varepsilon\}$.

Equation 3 is the formula to compute SIP. By unfolding this equation, we have:

$$Pr(q \subseteq g) = \sum_{i=1}^{|Eq|} (-1)^{i-1} \sum_{J \subseteq \{q_1,..,q_{|Eq|}\}, |J|=i} Pr(\wedge_{j=1}^{|J|} Bq_j). \quad (10)$$

Clearly, we need exponential number of steps to perform the exact calculation. Therefore, we use the threshold ϵ to stop the calculation as early as possible.

Lemma 2. *Let $S_i = \sum Pr(\wedge_{j=1}^{|J|} Bq_j)$ where J is an embedding subset, $J \subseteq \{q_1,..,q_{|Eq|}\}$ and $|J| = i$. [22] provides the bounds for* Inclusion-Exclusion Principle *such that,*

$$Pr(q \subseteq g) \begin{cases} \leq (-1)^{w-1} \sum_{w=1}^{i} S_w & \text{if } i \text{ is odd,} \\ \geq (-1)^{w-1} \sum_{w=1}^{i} S_w & \text{if } i \text{ is even.} \end{cases}$$

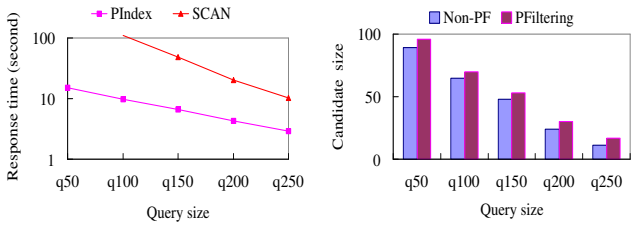
where $1 \leq i \leq |Eq|$, $1 \leq w \leq i$.

Based on these bounds and ϵ, we have the pruning rule:

Theorem 6. *When i is odd, if $(-1)^{w-1} \sum_{w=1}^{i} S_w < \epsilon$, g can be safely pruned. When i is even, if $(-1)^{w-1} \sum_{w=1}^{i} S_w \geq \epsilon$, g is in the final answers.*

5. PERFORMANCE EVALUATION

In this section, we report the effectiveness and efficiency test results of our new proposed techniques. Our methods are implemented on a Windows XP machine with a 3GHz Pentium IV CPU and 2GB main memory. Programs are compiled by Microsoft Visual C++ 6.0. In the experiments, we use a real uncertain graph database and a real deterministic graph database with synthetic uncertain information.



(a) Total query processing time (b) # Candidates after Prob. pruning

Figure 8: Scalability to # query size.

Real uncertain dataset. The real uncertain graph database was obtained from the STRING database[5] which has known and predicted protein interactions. We extract 1.5K uncertain graphs from the database. The uncertain graphs have an average number of 332 vertices and 584 edges. Vertices are deterministic, and each edge has an average value of 0.367 existence probability. Moreover, we generate labels for vertices with COG protein functions[6]. The total number of distinct vertex labels is 153. Each query set qi has 100 connected query graphs and query graphs in qi are connected size-i graphs (the edge number in each query is i), which are extracted from corresponding deterministic graphs of uncertain graphs randomly, such as $q50$, $q100$, $q150$, $q200$ and $q250$.

Real dataset with synthetic uncertain information. We generate uncertain information for AIDS antiviral screen dataset[7] (AIDS for short) that has been widely used in examination of deterministic graph search. We generate 10K connected and labeled graphs from the molecule structures and omit Hydrogen atoms. The *default database size* in experiments is 10K. The graphs have 24.3 vertices and 26.5 edges on average. A major portion of the vertices are C, O and N. The total number of distinct vertex labels is 62. We generate existence probability for vertices and edges following Gaussian distribution $N(\mu, \sigma)$. Using the same method in real uncertain graph dataset, We generate a query set, that is $q16$. The query set has 1K query graphs.

As introduced in Section 1.2, we implement the method in [29] to do structural pruning. This method is called *SFiltering* in experiments. We adopt the method in [32] to mine initial frequent features F_0. We select optimal features F from F_0 through a query log as follows: in each experiment, we divide a database set D to a query log set Γ ($\frac{4}{5}|D|$) and a testing query set q ($\frac{1}{5}|D|$). The method of probabilistic pruning is called *PFiltering* in experiments. Since there are no previous works on the topic studied in this paper, we compare the proposed algorithms with the naive method, *SCAN*, introduced in Section 1. The methods of exact calculation and bound of exact SIP in verification are called *E-BC* and *E-Bound* in experiments respectively. To compare with *PFiltering*, we feed initial features F_0 to our probabilistic pruning algorithm without optimal feature selection. This algorithm is called *Non-PF*. We report average results in following experiments.

5.1 Performance on Real Dataset

In this section, we present the performance evaluation on real uncertain graph datasets. The setting of experimental parameters is

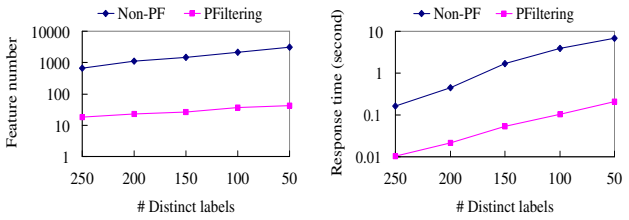
[5] http://string-db.org

[6] http://www.ncbi.nlm.nih.gov/COG/

[7] http://dtp.nci.nih.gov/

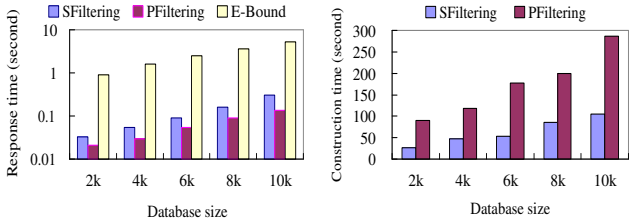
set as follows: the probability threshold is 0.3 - 0.7, and the default value is 0.5; the number of generated distinct labels is 50 - 200, and the default value is 150.

In the first experiment, we demonstrate the efficiency of *E-Bound* against *E-BC* in verification step. We first run structural and probabilistic filtering algorithms against the default real dataset to create candidate sets. The candidate sets are then verified for calculating SIP using proposed algorithms. Figure 5 reports the results, from which we know *E-Bound* is efficient under all parameter settings, while curves of *E-BC* grow in exponential. On average, the runtime of *E-Bound* is 20 times less than that of *E-BC*, since *E-Bound* can stop calculation as early as possible with tight probability bounds. We use *E-Bound* for verification in following experiments.



(a) # Features (b) Runtime

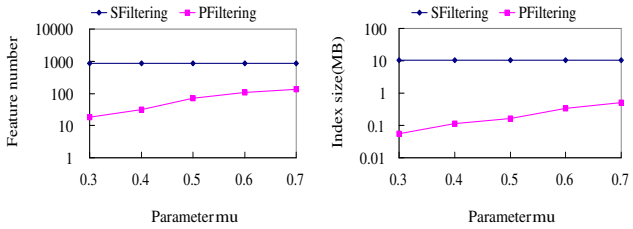
Figure 9: Scalability to # distinct labels.



(a) Query processing time (b) Construction time

Figure 10: Scalability to database size.

Figure 5.1 reports total query processing time and number of candidates after probabilistic pruning with respect to different query sizes. $PIndex$ denotes the entire algorithm, that is, a combination of *SFiltering*, *PFiltering* and *E-Bound*. All curves decrease with increasing of query size, as the probability of a large query graph being subgraph-isomorphic to g^c is low. From the result in Figure 8(a), we know that $PIndex$ has a very short runtime, while $SCAN$ takes much more time and has gone beyond 100 seconds at $q100$. The candidate size after probabilistic pruning is shown in Figure 8(b), from which we know that both *Non-PF* and *PFiltring* can lead to a very small size candidate set. *Non-PF* indexed all frequent features, and thus it has a good pruning power. This conclusion is confirmed in Figure 9(a), in which *Non-PF* has a large number of features. Also, as shown in this figure, *PFiltring* indexed a very small size feature set. However, *PFiltring* has almost the same number of candidate uncertain graphs as given in Figure 8(b). The results confirm that the selected optimal features have a very powerful pruning capability. Figure 9(b) gives pruning time for *Non-PF* and *PFiltring* with respect to distinct labels. As shown in the figure, *Non-PF* takes quite some time because a query graph needs an isomorphic test against its large feature set. *PFiltring* has efficient pruning time with average value less than 0.1 second.



(a) Varying μ (b) Varying μ

Figure 11: Impact of uncertainties on the efficiency.

5.2 Performance on Synthetic Dataset

This section gives the performance study on AIDS datasets. The setting of experimental parameters is set as follows: the probability threshold is 0.5; the database size (number of uncertain graphs in a database) is 2K - 10K, and the default value is 6K. The value of μ of Gaussian distribution is 0.3 - 0.7, and the default value is 0.5; the value of σ of Gaussian distribution is 0.1.

As important parameters such as size of the query graph, probability threshold and number of distinct labels are studied on the real dataset, we focus on investigating the scalability of the techniques with respect to the database size and uncertainties of data, in addition, the last section shows a comparison between our algorithm and a naive solution. In this section, we present evaluation of each step in our algorithm. Figure 5.1 reports the performance of techniques against the database size. Figures 10(a) and 10(b) show query processing time and construction time, for probabilistic and structural pruning, from which we see that all curves have good scalability under two metrics. As shown in Figure 10(b), the construction time of *PFiltering* is a little large due to the cost operations on examining the containment relationship between the initial feature set F_0 and query logs. However this process can select very powerful features for efficient search shown in Figure 10(a).

Figure 5.2 reported the impact of distributions of uncertainties on the efficiency with respect to varying μ from 0.3 to 0.7. The changes of μ result in a change of the average existence probability of vertices or edges. Both curves of *SFiltering* do not change, as it does pruning without consideration of uncertainties. From the figure, for *PFiltering*, we observe that both number and size of features increase, since the increase of μ leads to the decrease of the possibility of query satisfying CND. *PFiltering* has a quite small set of features, even the largest value is less than 120 when $\mu = 0.7$.

6. RELATED WORK

A systematic introduction to the topic of managing and mining uncertain data can be found in [2]. One related topic to this paper is probabilistic XML (PXML) that has been studied recently. Nierman et al. [23] proposed the Probabilistic Tree Data Base (ProTDB) to manage uncertain data represented in XML. Actually it belongs to the catalog of PrXML$^{\{ind,mux\}}$ model. Abiteboul et al. [27] propose a fuzzy tree model with nodes attached a conjunction of probabilistic event variables. In [19], Kimelfeld et al. systematically summarize the probabilistic XML models previously studied, the expressiveness and tractability of queries on different models, and give efficient algorithms for different data models. The above methods all focus on how to calculate probability efficiently over uncertain trees, which is relatively easy compared to calculation on uncertain graphs. In this paper, we not only consider efficient computations, but also consider probabilistic inverted index (PIndex) for pruning. The indexing idea of this paper can be expanded to PXML, with an improvement on computing efficiency.

Another related topic is the deterministic subgraph query. There are a lot of indexing and algorithms proposed for this problem. A major category is feature-based pruning, for example, GraphGrep [15], gIndex [33], FG-Index [7], Swift-Index [29], and etc. Another category are non-feature-based techniques, namely, Closure-Tree [16], gString [18] and GCoding [18]. These indexing methods cannot be applied to uncertain graphs.

With respect to uncertain graphs, Zou et al. [35, 36] study frequent subgraph mining on uncertain graph data under the probabilistic or expected semantics. Following their proposed uncertain graph models, we study the subgraph search problem in this paper. Potamias et al. [25] study k-nearest neighbor queries (k-NN) over uncertain graphs, i.e., computing the k closest nodes to a query node. They define an uncertain graph using possible world model, and propose three different probabilistic distance functions to define k-NN queries. They propose sampling algorithms to answer the #P-complete k-NN queries. The problem in [25] is different from ours, [25] emphasizes on how to efficiently compute probabilistic distance functions between two vertices, while ours focuses on calculating subgraph isomorphism probability. [17] studied path queries in uncertain road networks that are abstracted to uncertain graphs.

7. CONCLUSION

This is the first work to answering a threshold-based probability subgraph query over a large uncertain graph database. Though it is an NP-hard problem, we employ the filtering-and-verification methodology to answer the query efficiently. During the filtering phase, structural and probabilistic indices are used to filter out uncertain graphs as many as possible. For the probabilistic inverted index, an optimal feature selection strategy is given to maximize its pruning capability. The optimal strategy is proved to be NP-complete, thus, we propose a c-approximate algorithm to select a set of optimal features. During the verification phase, an exact algorithm with tight bounds is given to compute the final answer. Finally, we confirm our designs through an extensive experiments.

8. ACKNOWLEDGMENT

This research are supported by the National Science Fund for Distinguished Young Scholars (Grant No. 61025007), National Science Fund of China Key Program (Grant No. 60933001) and National Basic Research Program of China (973, Grant No. 2011CB302200-G).

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. *Managing and mining uncertain data*. Springer, 2009.

[3] E. Balas and J. Xue. Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. *Algorithmica*, 15:397–412, 1996.

[4] S. Beretti, A. Bimbo, and E. Vicario. Efficient matching and indexing of graph models in content based retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23:1089–1105, 2001.

[5] S. Biswas and R. Morris. Exor: opportunistic multi-hop routing for wireless networks. In *Proc. of SIGCOMM*, pages 101–122, 2005.

[6] C. Chen, X. Yan, P. S. YuE, J. Han, D.-Q. Zhang, and X. Gu. Towards graph containment search and indexing. In *Proc. of VLDB*, 2007.

[7] J. Cheng, Y. Ke, and W. Ng. Efficient query processing on graph databases. *ACM Trans. on Database Systems (TODS)*, 34(1), 2009.

[8] H. Chui, W.-K. Sung, and L. Wong. Exploiting indirect neighbours and topological weight to predict protein function from protein-rotein interactions. *Bioinformatics*, 22(13):47–58, 2007.

[9] L. P. Cordellaand, P. Foggia, and C. Sansone. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(10):1367–1372, 2004.

[10] C.Wang and L. Chen. Continuous subgraph pattern search over graph streams. In *Proc. of ICDE*, pages 872–883, 2009.

[11] N. N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *Proc. of PODS*, pages 223–232, 2007.

[12] P. Domingos and M. Richardson. Mining the network value of customers. In *Proc. of KDD*, pages 356–365, 2001.

[13] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W.H.Freeman, 1979.

[14] J. Ghosh, H. Ngo, S. Yoon, and C. Qiao. On a routing problem within probabilistic graphs and its application to intermittently connected networks. In *Proc. of INFOCOM*, pages 1768–1779, 2007.

[15] R. Giugno and D. Shasha. Graphgrep: a fast and universal method for querying graphs. In *Proc. of the International Conference on Pattern Recognition*, pages 112–115, 2002.

[16] H. He and A. K. Singh. Closure-tree: an index structure for graph queries. In *Proc. of ICDE*, pages 426–437, 2006.

[17] M. Hua and J. Pei. Probabilistic path queries in road networks: traffic uncertainty aware path selection. In *Proc. of EDBT*, pages 347–358, 2010.

[18] H. Jiang, H. Wang, P. S. Yu, and S. Zhou. Gstring: a novel approach for efficient search in graph databases. In *Proc. of ICDE*, pages 566–575, 2007.

[19] B. KIMELFELD, Y. KOSHAROVSKY, and Y. SAGIV. Query efficiency in probabilistic xml models. In *Proc. of SIGMOD*, pages 776–787, 2008.

[20] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proc. of CIKM*, pages 87–96, 2003.

[21] B. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(3):493–504, 1998.

[22] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[23] A. Nierman and H. V. Jagadish. Protdb: probabilistic data in xml. In *Proc. of VLDB*, pages 1476–1487, 2002.

[24] E. Petrakis and C. Faloutsos. Similarity searching in medical image databases. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(3):435–447, 1997.

[25] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios. k-nearest neighbors in uncertain graphs. In *Proc. of VLDB*, pages 863–874, 2010.

[26] S. Rintaro, S. Harukazu, and H. Yoshihide. Interaction generality: a measurement to assess the reliability of a protein-protein interaction. *Nucleic Acids Research*, 30(5):1163–1168, 2002.

[27] P. Senellart and S. Abiteboul. On the complexity of managing probabilistic xml data. In *Proc. of PODS*, pages 283–292, 2007.

[28] P. Seshadri and A. N. Swami. Generalized partial indexes. In *Proc. of ICDE*, pages 375–386, 1995.

[29] H. Shang, Y. Zhang, X. Lin, and J. X. Yu. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. In *Proc. of VLDB*, pages 364–375, 2008.

[30] D. Suciu and N. N. Dalvi. Foundations of probabilistic answers to queries. In *Proc. of SIGMOD*, pages 274–285, 2005.

[31] S. Suthram, T. Shlomi, E. Ruppin, R. Sharan, and T. Ideker. A direct comparison of protein interaction confidence assignment schemes. *Bioinformatics*, 7(1):360, 2006.

[32] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Proc. of ICDM*, pages 721–724, 2002.

[33] X. Yan, P. S. Yu, and J. Han. Graph indexing: a frequent structurebased approach. In *Proc. of SIGMOD*, pages 335–346, 2004.

[34] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *Proc. of CVPR*, pages 34–41, 2008.

[35] Z. Zou, H. Gao, and J. Li. Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In *Proc. of KDD*, pages 633–642, 2010.

[36] Z. Zou, H. Gao, and J. Li. Mining frequent subgraph patterns from uncertain graph data. TKDE, volume 22, pages 1203–1218, 2010.

Symbol	Description
D, SC_q, C_q, A_q	the uncertain database
D^c, SC_q^c	the deterministic database
D_f	the inverted list
g	the uncertain graph
ϵ	the user-specified probability threshold
f, q, g' g^c	the deterministic graph
$Lower(f), Upper(f)$	the lower and upper bounds of $Pr(q \subseteq g)$
Bq_i, Bf_i	the boolean variable of an embedding
Ef, Eq	the set of embeddings
$SI(q, g)$	the possible world graph set
F_0, F	the feature set
CND	the probabilistic pruning condition
$SCAN$	the naive method
$Pr(q \subseteq g)$	the subgraph isomorphism probability between q and g

Table 1: Notations.

APPENDIX

A. PROOFS AND ALGORITHMS

A.1 Proof of Theorem 2

Proof. Consider an uncertain graph g with deterministic vertices ($P_V = 1$) and uncertain edges.

Firstly, we define *query cut*: For a query q, a query cut is a set of edges in g^c, and the removing of the set of edges leads to the absence of all q's embeddings in g^c. A query cut is minimal if no proper subset of the query cut is a query cut. Here, we assume that a query cut is minimal.

Next, we assume that all m edges of g have an inexistence probability $1 - p$. Then, based on Definition 4, we have

$$Pr(q \subseteq g) = 1 - \sum_{i=0}^{m} \lambda_i (1-p)^i p^{m-i},$$

where λ_i is the number of query cuts of cardinality i, and thus

$$\sum_{i=0}^{m} \lambda_i (\frac{1-p}{p})^i = p^{-m}(1 - Pr(q \subseteq g)). \quad (11)$$

Suppose we could compute $Pr(q \subseteq g)$ efficiently for an arbitrary value p. By substituting $m+1$ distinct values of p into Equation (11), we get a system of $m+1$ linear equations in $m+1$ unknowns λ_i with known right-hand sides. The coefficient matrix of this system is Vandermonde matrix with nonzero determinant, and hence we can solve for the λ_i values in polynomial time. In particular, we could find λ_i for the smallest size of the query cut in g^c. Therefore an efficient method for evaluating $Pr(q \subseteq g)$ would yield an efficient method for computing the number of minimum cardinality query cuts.

Finally, we build a connection between query cuts in g^c and cut sets for two vertices in a deterministic graph. Suppose q has Eq embeddings in g^c, and each embedding has k edges. Assign k labels, $\{e_1, ..., e_k\}$, for edges of each embedding (the order is random.). Create a corresponding *line* graph for each embedding by (1) create $k+1$ *isolated* nodes, and (2) connect these $k+1$ nodes to be a line by associating k edges (with corresponding labels) of the embedding. Based on these line graphs, we construct a *parallel* graph, cG. The node set of cG consists of all nodes of the Eq line graphs and two new nodes, s and t. The edge set of cG consists of all edges (with labels) of the Eq line graphs. In addition, one edge (without label) is placed between an end node of each line graph and s. Similarly, there is an edge between t and the other end node of each line graph. As a result, Eq embeddings are transformed into a deterministic graph cG. Based on this transformation, it is easy to have the following lemma.

Lemma 3. *The query cut set of g^c is also the cut set (without edges incident to s and t) from s to t in cG.*

Based on this lemma, we get the following corollary.

Corollary 1. *Computing the number of minimum cardinality query cuts in g^c equals determining the number of minimum cardinality cuts between s and t in cG.*

Recall that we can compute the number of minimum cardinality query cuts in polynomial time. However, the problem of computing the number of minimum cardinality $s-t$ cuts is #P-complete [13], which leads to a contradiction. ■

A.2 Proof of Lemma 1

Proof. From Definition 4, we have

$$Pr(q \subseteq g) = \sum_{g' \in SI(q,g)} Pr(g \Rightarrow g') \quad (12)$$

where $SI(q, g)$ is g's possible world graphs that are subgraph isomorphic to q. We divide $SI(q, g)$ into $|Eq|$ subsets such that a possible world graph in SI_i contains i embeddings where $1 \leq i \leq |Eq|$. Thus, from Equation 12, we get

$$\begin{aligned} Pr(q \subseteq g) &= \sum_{g' \in SI_1 \cup \ldots \cup SI_{|Eq|}} Pr(g \Rightarrow g') \\ &= \sum_{1 \leq j_1 \leq |Eq|} \sum_{g' \in SI_{j_1}} Pr(g \Rightarrow g') - \sum_{1 \leq j_1 < j_2 \leq |Eq|} \sum_{g' \in SI_{j_1} \cap SI_{j_2}} Pr(g') \\ &+ \cdots + (-1)^{i-1} \sum_{1 \leq j_1 < \ldots < j_i \leq |Eq|} \sum_{g' \in SI_{j_1} \cap \ldots \cap SI_{j_i}} Pr(g \Rightarrow g') + \cdots \\ &+ (-1)^{|Eq|-1} \sum_{g' \in SI_{j_1} \cap \ldots \cap SI_{j_{|Eq|}}} Pr(g \Rightarrow g'). \end{aligned} \quad (13)$$

Consider the ith item on the RHS in Equation 13. Let A be a subgraph of g^c, composed of i embeddings, and $B = Bq_{j_1} \wedge ... \wedge Bq_{j_i}$ be the corresponding boolean variable of A. The set $g' \in SI_{j_1} \cap ... \cap SI_{j_i}$ contains all PWGs that have A as a subgraph. Then, for the ith item, we get,

$$\begin{aligned} &(-1)^{i-1} \sum_{1 \leq j_1 < \ldots < j_i \leq |Eq|} \sum_{g' \in SI_{j_1} \cap \ldots \cap SI_{j_i}} Pr(g \Rightarrow g') \\ &= (-1)^{i-1} \sum_{1 \leq j_1 < \ldots < j_i \leq |Eq|} Pr(Bq_{j_1} \wedge ... \wedge Bq_{j_i}). \end{aligned} \quad (14)$$

Similarly, we can get the results for other items. By replacing the corresponding items with these results in Equation 13, we get

$$\begin{aligned} Pr(q \subseteq g) &= \sum_{1 \leq j_1 \leq |Eq|} Pr(Bq_j) - \sum_{1 \leq j_1 < j_2 \leq E|} Pr(Bq_{j_1} \wedge Bq_{j_2}) \\ &+ \cdots + (-1)^{i-1} \sum_{1 \leq j_1 < \ldots < j_i \leq |Eq|} Pr(Bq_{j_1} \wedge ... \wedge Bq_{j_i}) \\ &+ \cdots + (-1)^{|Eq|-1} Pr(Bq_{j_1} \wedge ... \wedge Bq_{j_{|Eq|}}). \end{aligned} \quad (15)$$

Based on the *Inclusion-Exclusion Principle* [22], the RHS of Equation 15 is just $Pr(Bq_1 \vee ... \vee Bq_{|Eq|})$. ■

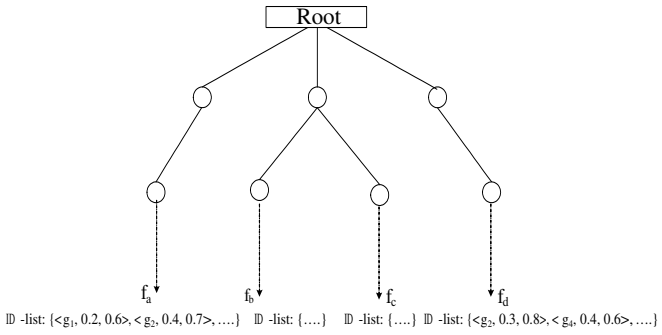


Figure 12: Prefix-tree structure of probabilistic inverted index.

A.3 Proof of Theorem 3

Proof. From Lemma 1, it is easy to see

$$Pr(q \subseteq g) = Pr(Bq_1 \vee ... \vee Bq_{|Eq|}),$$
$$Pr(f \subseteq g) = Pr(Bf_1 \vee ... \vee Bf_{|Ef|}).$$

where $Ef = \{f_1, ..., f_{|Ef|}\}$ is the set of all embeddings of f in g^c.

Since $q \supseteq f$, we have $Bq_1 \vee ... \vee Bq_{|Eq|} \subseteq Bf_1 \vee ... \vee Bf_{|Ef|}$. Also, based on $UpperB(f) < \epsilon$, we obtain $Pr(q \subseteq g) \leq Pr(f \subseteq g) \leq UpperB(f) < \epsilon$. Then g can be pruned. ■

A.4 Proof of Theorem 4

Proof. Similar as proof in Theorem 3, we can show that $Pr(q \subseteq g) \geq Pr(f \subseteq g) \geq LowerB(f) \geq \epsilon$. Then $g \in A_q$. ■

A.5 Proof of Theorem 5

Proof. Based on possible world semantics, we can obtain

$$Pr(Bf_i) = \sum_{g' \in \Omega_f} Pr(g \Rightarrow g'). \quad (16)$$

where Ω_f is the possible worlds that contain f as a subgraph.

By substituting Equation 1 into Equation 16, Equation 16 is written as

$$Pr(Bf_i) = \sum_{g' \in \Omega_f} \prod_{v \in g'} P_V(v) \prod_{v \in g \setminus g'} (1 - P_V(v)) \prod_{e=(u,v) \in g'} P_E(e|u,v) \prod_{e=(u,v) \in g \setminus g'} (1 - P_E(e|u,v)).$$

Since g' contains f, we obtain

$$Pr(Bf_i) = \prod_{v \in f} P_V(v) \prod_{e=(u,v) \in f} P_E(e|u,v) \sum_{g' \in \Omega_f} \prod_{v \in g' \setminus f} P_V(v) \prod_{v \in g \setminus g'} (1 - P_V(v)) \prod_{e=(u,v) \in g' \setminus f} P_E(e|u,v) \prod_{e=(u,v) \in g \setminus g'} (1 - P_E(e)). \quad (17)$$

Observe that the item $\sum_{g' \in \Omega_f}(\cdot)$ in the RHS of Equation 17 equals 1. Thus, Equation 6 holds. ■

A.6 Probabilistic Inverted Index

A.6.1 Indexing Model

To create a prefix-tree structured index, for each feature $f \in F$, we first adopt the method in [29] to construct a sequence *QI-Sequence*, SEQ_f, that is a string with structural information of f. Then, based on strings of features, we construct a prefix-tree in which the path from root to leaf node represents the string of a feature. The inverted list D_f of each feature is attached to the leaf node of its corresponding path. Figure 12 shows a prefix-tree structure.

Based on prefix-tree, we have the following pruning condition: *Prefix-pruning*[29]. If a prefix SEQ_f^i of SEQ_f cannot be mapped to a query graph q's QI-Sequence, then SEQ_f of f cannot be mapped to a query graph q's QI-Sequence. In other words, we can prune away the feature f as long as we find its prefix SEQ_f^i does not map to a query q.

Given a query graph q and the prefix-tree, the process of finding all features in the prefix-tree that are contained in q is conducted by traversing the prefix-tree from the top to the bottom in a depth-first fashion. When visiting a node in the prefix-tree, n_i, the path from the root to the node n_i represents SEQ_f^i for all tree features SEQ_f that have SEQ_f^i as their prefix. If SEQ_f^i is not sub-isomorphic to q, then there is no need to further examine the subtrees below n_i in the prefix-tree.

The prefix-pruning can save many isomorphism tests. Consider pruning condition 1: $q \supseteq f \wedge UpperB(f) < \epsilon$. Based on prefix-pruning condition, it is easy to see that we can save more cost than the sequence scan for $q \supseteq f$. Therefore, the prefix-pruning also saves cost for pruning 1 based on the observation that the cost of operation on $UpperB(f) < \epsilon$ is negligible compared to operation on $q \supseteq f$. Consider pruning condition 2: $q \subseteq f \wedge LowerB(f) \geq \epsilon$. The prefix-pruning condition cannot prune away query before the query traverses to a leaf node of the prefix-tree due to condition $q \subseteq f$. However, for a query q, pruning 2 only returns a set of final answers whose size is much smaller than that of graph set filtered out by pruning condition 1. Therefore, the prefix-tree structure can lead to a much more efficient pruning than the flat structure.

A.6.2 Index Maintenance

The indexing algorithm we have proposed so far is suitable in a static scenario. When updates take place in the database D or query graphs deviate away from previously logged entries, we take following steps:

We keep the same set of selected optimal features and the same prefix-tree built. Whenever an update to D takes place, we simply update a corresponding entry of probabilistic index. If the changed uncertain graph database or query log is not significantly deviated from the original one, it is reasonable that the maintained index will continue to perform well. Otherwise, we periodically take small samples from uncertain databases and query log, mine a set of frequent subgraphs out of uncertain graph samples, and calculate a new index based on the samples. The mining step can be bypassed, because frequent patterns represent the intrinsic trends of data and are relatively stable in spite of updates. The sampling ratios can be set according to the updating rates of databases and query log.

The above update method relates to statistical information of query log. Here the global probabilistic matrix **M** is used to collect the statistical information. In **M**, we record the number of non-zero entries for each feature, so that we can construct a histogram for the initial feature set F_0. Recall that optimal features are selected based on the non-zero entries in **M**. Thus the histogram gives accurate statistical information for query log. Another issue is that we should efficiently monitor a deviation of the query log, so that we can maintain PIndex flexibly. In this paper, we adopt the approach proposed in [10] for efficiently querying graph steams to monitor graph query logs. In uncertain graph applications [31, 8, 26], the updating rate of query log is much slower than that of graph steam [10]. Therefore, [10] is efficient enough to handle graph query logs.

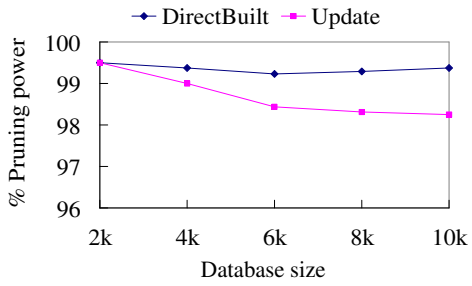


Figure 13: Performance of the index maintenance algorithm.

Figure 13 shows the performance of the index maintenance algorithm. In the experiment, we compare two curves: one curve shows the performance of the index directly constructed for each database ($DirectBuilt$), while the other shows the performance of the index originally constructed for database with 2K size but maintained afterwards using the index maintenance algorithm ($Update$). The two curves represent *pruning power* of probabilistic pruning, where pruning power is defined as $\frac{|UC_q|-|C_q|}{|UC_q|-|A_q|}$. It can be seen that two curves start from the same point, and keep quite close to each other for all the following updated databases. This result validates the feasibility of the index maintenance scheme.

A.7 Upper and Lower Bounds of SIP

We show values and calculation time of upper and lower bounds of subgraph isomorphism probability (SIP) of features mined from the deterministic graphs of uncertain ones, which are randomly extracted from the STRING database (http://string-db.org). On average, each graph has 330 vertices and 584 edges, and the existence probability of each edge is 0.376. We mined frequent features from corresponding deterministic graphs and calculated exact SIP probabilities ($Exact$), lower bound ($LowerBound$) and upper bound ($UpperBound$) of SIPs of the features using methods proposed in this subsection.

Figure 14 shows the average values of the bounds. From the figure, we can observe that $UpperBound$ and $LowerBound$ come very close to the exact SIP, which indicates the bounds obtained from the proposed methods are very tight. Figure 15 shows that $UpperBound$ and $LowerBound$ can be computed quite efficiently. Moreover, the result confirms that the time needed to compute the $exact$ value is much larger than that to compute the bounds.

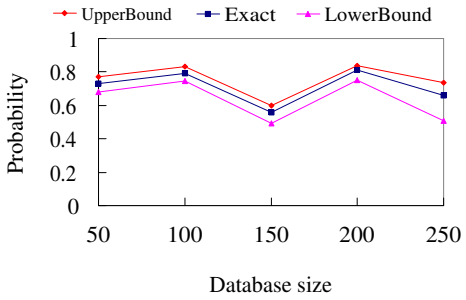


Figure 14: Upper and lower bounds of exact SIP of features.

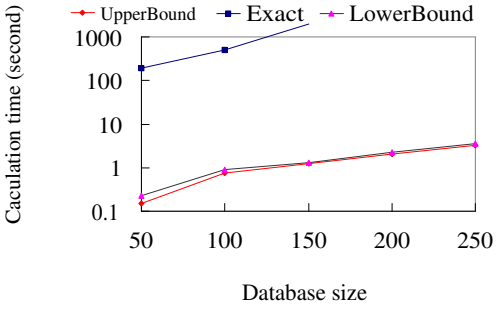


Figure 15: Calculation time of upper and lower bounds.