Efficient Probabilistic Supergraph Search over Large Uncertain Graphs

Yongxin Tong Xiaofei Zhang Caleb Chen Cao Lei Chen
Hong Kong University of Science & Technology, Hong Kong, China
{yxtong, zhangxf, caochen, leichen}@cse.ust.hk

ABSTRACT

In recent years, with the emergence of a number of new real applications, such as protein-protein interaction (PPI) networks, visual pattern recognition, and intelligent traffic systems, managing huge volumes of uncertain graphs has attracted much attention in the database community. Currently, most existing fundamental queries over graphs only support deterministic (or certain) graphs, although real graph data are often noisy, inaccurate, and incomplete. In this paper, we study a new type of uncertain graph query, *probabilistic supergraph containment query* over large uncertain graphs. Specifically, given an uncertain graph database UGD which contains a set of uncertain graphs, a deterministic query graph q, and a probabilistic threshold δ, *a probabilistic supergraph containment query* is to find the set of uncertain graphs from UGD, denoted as UGD_q, such that $UGD_q = \{ug_i \in UGD | Pr(ug_i \subseteq q) \geq \delta\}$ where $Pr(ug_i \subseteq q)$ means the likelihood that ug_i is a subgraph of q. We prove that the computation of $Pr(ug_i \subseteq q)$ is #P-hard and design an efficient filtering-and-verification framework to avoid the expensive computation. In particular, we propose an effective filtering strategy and a novel probabilistic inverted index, called *PS-Index*, to enhance pruning power in the filtering phase. Furthermore, the candidate graphs which pass the filtering phase are tested in the verification phase via an efficient unequal probability sampling-based approximation algorithm. Finally, we verify the effectiveness and efficiency of the proposed methods through extensive experiments.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems-*Query processing*

General Terms

Algorithm, Experimentation, Performance

Keywords

Uncertain Graphs, Graph Querying, Probabilistic Supergraph Query

1. INTRODUCTION

Due to the emergence of many real-world applications of uncertain graphs, such as protein-protein interaction (PPI) network analysis [7, 15, 17], visual pattern recognition [2, 3], social network mining [1], and intelligent traffic monitoring [9], etc., query processing over large uncertain graphs has attracted much attention in the database community. Several uncertain graph queries, probabilistic subgraph (or subgraph similarity) queries [24, 23], k nearest neighbor (kNN) query over uncertain graphs [13], uncertain shortest path query [9], and uncertain distance-constraint reachability query [10], have been studied recently.

In this paper, we investigate another fundamental graph query, *supergraph containment query*, in large uncertain graphs. As a motivating example for this query, we consider the case in PPI networks where existences of some interactions are defined with uncertainty. Specially, in a PPI network[7, 17], vertices represent proteins, edges represent interactions between proteins, the labels of vertices are the COG functional annotations of proteins. It is reasonable to model a PPI network as an uncertain graph [23, 24, 13, 27, 26, 28]. In particular, when researchers study some new types of PPI structures, they want to efficiently identify what new PPI structures can be included by some known important PPI structures (which may represent important PPI structural properties) with high probability. Therefore, it is essential to study the containment relationship where some uncertain graphs are contained into a given query graph.

The problem of supergraph containment query over deterministic graphs is defined as follows [5]: Given a deterministic graph database, $GD = \{g_1, \ldots, g_n\}$ and a query graph q, it is to find the answer set $GD_q = \{g_i \in GD | g_i \subseteq q\}$, where $g_i \subseteq q$ means that g_i is a subgraph of q. The containment relationship between a query graph and each deterministic graph is determined by subgraph isomorphism test, which is a NP-complete problem. Compared with deterministic graphs, it is much harder for uncertain graphs to decide such a containment relationship between a query graph and each uncertain graph due to the inherent uncertainty. According to the widely adopted possible world semantic over uncertain graphs [23, 24, 27, 28], each possible world graph of a given uncertain graph is a possible instance of the uncertain graph. Therefore, the problem of probabilistic supergraph containment query over uncertain graphs is described as follows. Given an uncertain graph database, $UGD = \{ug_1, \ldots, ug_n\}$, a query graph q, and a probabilistic threshold δ ($0 < \delta \leq 1$), this query is to find all uncertain graphs $ug_i \in UGD$ such that $Pr(ug_i \subseteq q) \geq \delta$, where $Pr(ug_i \subseteq q)$ is the *supergraph containment probability (SCP)* between q and ug_i. In reality, *SCP* is equal to the sum of the probabilities of all the possible world graphs that are contained by q. We will illustrate this problem via the following example.

EXAMPLE 1 (*Probabilistic Supergraph Containment Query*). *Figure 1 shows an uncertain graph database including four uncertain graphs. The vertices are labeled from a letter set, A, B, C, D (in this example, we ignore labels of edges for the succinct*

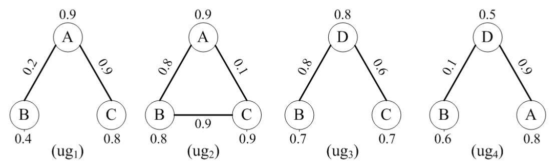


Figure 1: An uncertain graph database

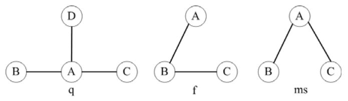


Figure 2: A query graph and a feature graph

reason as it is easy to extend labels on edges). In addition, each vertex and each edge are associated with a real number indicating their existence probabilities. As an example, all possible world graphs of ug_2 are shown in Figure 3 where every real number below each possible world graph represents the existence probability of such a possible world graph. Since each possible world graph is an instantiation of an uncertain graph, the sum of probabilities of 18 possible world graphs of ug_2 equals to 1. Given the probabilistic threshold δ=0.7, the SCP between q (as shown in Figure 2) and ug_2 can be calculated as follows: $Pr(ug_2 \subseteq q)$=$Pr(pw_1) + Pr(pw_2) + Pr(pw_3) + Pr(pw_4) + Pr(pw_5) + Pr(pw_6) + Pr(pw_7) + Pr(pw_8) + Pr(pw_9) + Pr(pw_{11}) + Pr(pw_{12}) + Pr(pw_{13}) + Pr(pw_{15}) = 0.352 < 0.7$. Thus, ug_2 cannot be returned as a final result. Similarly, we can also check ug_1, ug_3 and ug_4 via the same method.

Based on the aforementioned problem statement, a naive solution is to sequentially scan all uncertain graphs and calculate their *SCP* one by one. It is obviously inefficient. In addition, a closely related work, targeting at probabilistic subgraph search over uncertain graphs, was proposed recently [24]. This work designs an efficient search framework in three steps: structural pruning, probabilistic pruning and verification. For an uncertain graph database, this method first neglects all probabilities of uncertain graphs and uses existing techniques of subgraph search in deterministic graphs to prune some uncertain graphs. Then, it further uses probabilistic pruning to filter more candidates. Finally, it refines the final result in the verification phase. Unfortunately, we cannot adopt the aforementioned framework in [24] since there are intrinsic differences between subgraph search and supergraph search over uncertain graphs. Here, we firstly define several notations for the convenience of explanation.

We denote the uncertain graph database as UGD, and its corresponding deterministic graph database as GD, which is obtained by removing all the existence probability values in UGD. Moreover, their query results with the same query graph q are denoted as UGD_q and GD_q, respectively. For subgraph search over uncertain graphs, we usually observe $|UGD_q| \le |GD_q|$, where $|UGD_q|$ and $|GD_q|$ mean the sizes of UGD_q and GD_q. On the contrary, $|UGD_q| \ge |GD_q|$ in supergraph search over uncertain graphs. For example, if an edge, AB which includes two vertices A and B, is a query graph in the subgraph query, the probabilistic threshold $\delta = 0.5$, $GD_{AB} = \{ug_1, ug_2\}$ and $UGD_{AB} = \{ug_1\}$, respectively. Thus, $|GD_{AB}| = 2 > 1 = |UGD_{AB}|$ in Figure 1. In contrast to the subgraph query, given the supergraph query q in Figure 2, $GD_q = \{ug_1\}$ and $UGD_q = \{ug_1, ug_4\}$ with respect to $\delta = 0.5$. In other words, $|GD_q| = 1 < 2 = |UGD_q|$ for the supergraph queries in Figure 1. Therefore, when $|UGD_q| \le |GD_q|$, it is reasonable to firstly obtain the candidate set by the structural pruning techniques used in the deterministic subgraph search then further filter the candidates via the probabilistic pruning methods.

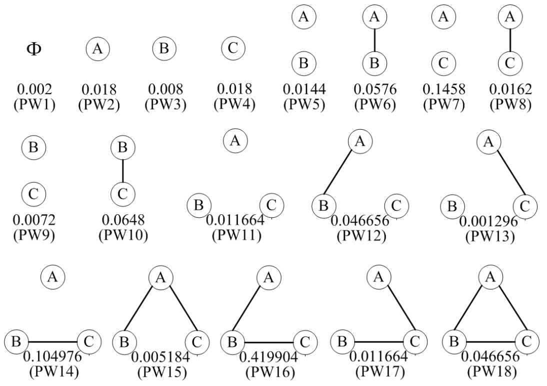


Figure 3: All Possible World Graphs of the uncertain graph ug_2

However, when $|UGD_q| \ge |GD_q|$, we cannot use the result set of deterministic supergraph search as the candidate set because uncertain graphs may be over-pruned in the structural pruning phase. Therefore, the structural pruning pruning adopted in [24] cannot be applied to solve our problem.

From the above discussion, we discover a probabilistic supergaph filtering strategy. Before introducing the filtering strategy, we first define the concept of *feature graphs*, which is a deterministic graph and a subgraph of at least a deterministic graph in GD. Based on feature graphs, the filtering strategy is, given a feature subgraph f, if $f \nsubseteq q$ and $Pr(f \subseteq ug_i) = p$, $Pr(ug_i \subseteq q) \le 1 - p$, where $Pr(f \subseteq ug_i)$ is the subgraph isomorphism probability between f and ug_i, which is the sum of the probabilities of all possible world graphs of ug_i that contains q. In other words, if $f \nsubseteq q$ and $p \le \delta$, ug_i can be pruned. A formal descriptions and proofs of above properties will be given in Section 3. We illustrate the above filtering strategy with the following example.

EXAMPLE 2 (*Probabilistic Supergraph Filtering Strategy*). *Given an uncertain graph database in Figure 1, a query graph q in the left of Figure 2, a probabilistic threshold δ=0.7, a feature subgraph f in the right of Figure 2, and all possible world graphs of ug_2 in Figure 3, based on the aforementioned filtering strategy, we can firstly find $f \nsubseteq q$, and $Pr(f \subseteq ug_2) = Pr(pw_{16}) + Pr(pw_{18}) = 0.467 > 0.3 = 1 - 0.7$. Thus, ug_2 is pruned.*

According to the aforementioned filtering strategy, it is still nontrivial to decide an optimal feature graph that best serves the pruning efficiently. In this work, we further propose an effective probabilistic inverted index, *PS-Index*. In addition, in order to speed up the verification step, we design an efficient sampling algorithm to calculate SCP. To sum up, we make the following contributions:

- We propose a new problem, the probabilistic supergraph containment query, and prove that it is a #P-hard problem.
- We design a probabilistic supergaph filtering strategy, a novel probabilistic inverted index, and an unequal-probability-sampling-based approximation algorithm in the verification phase.
- We verify the effectiveness and efficiency of the proposed solutions with extensive experiments.

The rest of the paper is organized as follows. Our problem formulation and complexity analysis are introduced in Section 2. In Section 3, we propose our solution framework toward the probabilistic supergraph containment query. Based on our probabilistic supergaph filtering logic, an optimal feature selection approach and a novel index, *PS-Index* which help eliminate redundant computation, are elaborated in Section 4. An unequal-probability-sampling-based approximate algorithm in the verification phase is shown in Section 5. Experimental studies and related work are reported in Section 6 and Section 7, respectively. Finally, we conclude in Section 8.

Notation	Meaning
UGD	an uncertain graph database
ug	an uncertain graph in UGD
PW	a set of all possible worlds generated from UGD
$pw_i(ug)$	the i-th possible world graph of ug
q	a query graph
F	a feature set where each feature is a deterministic graph
f_i	the i-th feature in F
$Pr(q \subseteq ug)$	subgraph isomorphism probability that q is contained ug
$Pr(ug \subseteq q)$	supergraph containment probability that q contains ug

Table 1: Summary of Notations

2. PROBLEM DEFINITION

In this section, we first review preliminaries with respect to deterministic graph search in Section 2.1. Then, we introduce our uncertain graph model and problem statement in Section 2.2. In addition, a summary of notations is shown in Table 1.

2.1 Preliminaries

DEFINITION 1. *(Deterministic Graph) A labeled undirected graph is denoted as $g =< V, E, L_v, L_e >$, where 1) V is the set of vertices; 2) E is the set of edges; 3) L_v is a label function assigning labels to each vertex in V; 4) L_e is a label function assigning labels to each edge in E.*

DEFINITION 2. *(Subgraph Isomorphism) Given two graphs, $g =< V, E, L_v, L_e >$ and $g' =< V', E', L'_v, L'_e >$, a subgraph isomorphism from g to g' is an injective function $f : V \to V'$, such that 1)$\forall u \in V$, $L_v(u) = L'_v(f(u))$; 2)$\forall (u, v) \in E$, $(f(u), f(v)) \in E'$ and $L_e(u, v) = L'_e(f(u), f(v))$.*

DEFINITION 3. *(Subgraph and Supergraph) Given two graphs, g and g', if there is a subgraph isomorphism from g to g', g is a subgraph of g', denoted as $g \subseteq g'$, and g' is called a supergraph of g.*

DEFINITION 4. *(Subgraph Query) Given a deterministic graph database $GD = \{g_1, \dots, g_n\}$ and a query graph q, the problem of subgraph query is to find the query answer set $D_q = \{g_i \in GD | q \subseteq g_i\}$.*

DEFINITION 5. *(Supergraph Containment Query) Given a deterministic graph database $GD = \{g_1, \dots, g_n\}$ and a query graph q, the problem of supergraph containment query is to find the query answer set $D_q = \{g_i \in GD | g_i \subseteq q\}$.*

2.2 Probabilistic Supergraph Containment Query

In this subsection, we first introduce the uncertain graph model based on the possible world semantics. Then, we define several important concepts in uncertain graph query, such as the subgraph isomorphism probability and the supergraph containment probability. Finally, we give the problem statement and analyze the complexity of this problem.

DEFINITION 6. *(Uncertain Graph) Given a deterministic graph $g =< V, E, L_v, L_e >$, a corresponding uncertain graph is defined $ug =< V, E, L_v, L_e, P_v, P_e >$, where 1) $P_v : V \to [0, 1]$ is an injective function assigning existence probabilities to each vertex in V; 2) $P_e : E \to [0, 1]$ is also an injective function assigning existence probabilities to each edge in E (where existence probabilities of edges assigned by P_e are actually conditional probabilities when endpoints of edges exist).*

DEFINITION 7. *(Possible World Graph) Given an uncertain graph $ug =< V, E, L_v, L_e, P_v, P_e >$, a possible world graph of ug is an instance of ug and is denoted $pw(ug) =< V', E', L'v, L'e >$, where 1) $V' \subseteq V$; 2) $E' \subseteq E \cap (V' \times V')$; 3) L'_v is a label function assigning labels to V'; 4) L'_e is a label function assigning labels to E'. L'_v and L'_e may assign the same labels of L_v and L_e or the non-existence if the corresponding vertex and edge do not exist in this instance*

Similar to previous studies of uncertain graph query [23, 24, 26, 27, 28], we also assume that both the existence probabilities of vertices and the conditional probabilities of edges of an uncertain graph are mutually independent. Thus, we can compute the probability of a possible world graph of a uncertain graph ug, denoted $Pr(pw(ug))$, as follows.

$$Pr(pw(ug)) = \prod_{v \in V'} Pr(v) \times \prod_{v \in V \setminus V'} (1 - Pr(v)) \times \prod_{e=(u,v) \in E'} Pr(e|u, v) \times \prod_{e=(u,v) \in E \cap (V' \times V') \setminus E'} (1 - Pr(e|u, v)) \quad (1)$$

In addition, the set of all possible worlds implicated from UGD is denoted as PW in this paper. Based on the aforementioned definitions, we introduce the concept of the supergraph containment probability and the problem statement as follows.

DEFINITION 8. *(Supergraph Containment Probability (SCP)) Given an uncertain graph ug and a deterministic query graph q, the supergraph containment probability between q and ug is defined:*

$$Pr(ug \subseteq q) = \sum_{pw_i(ug) \subseteq q \cap pw_i(ug) \in PW(ug)} Pr(pw_i(ug)) \quad (2)$$

where $PW(ug)$ means the set all possible world graphs generated from ug, $pw_i(ug) \subseteq q$ and $pw_i(ug) \in PW(ug)$ mean that possible world graphs which are contained by q.

Problem Statement (Probabilistic Supergraph Containment Query over Uncertain Graph Databases): *Given an uncertain graph database $UGD = \{ug_1, \dots, ug_n\}$, a deterministic query graph q and a probabilistic threshold δ, this problem is to find the query answer set $UGD_q = \{ug_i \in UGD | Pr(ug_i \subseteq q) \geq \delta\}$.*

Based on the problem statement, a naive solution is to scan each uncertain graph in UGD and compute the *SCP* one by one. Unfortunately, as shown in the following theorem, the problem of computing the SCP is #P-hard.

THEOREM 1. *(The Complexity of Supergraph Containment Probability (SCP) Computation) It is a #P-hard to compute the supergraph containment probability.*

Proof (Sketch): In order to prove the problem of computing the supergraph containment probability is #P-hard, we reduce it from the monotone DNF counting problem (#MDNF), which is known to be #P-complete [20].

Consider an instance of #MDNF: Given a monotone DNF formula $F = C_1 \vee \cdots \vee C_n$, with n clauses and m Boolean variables $v_1, \dots, v_m$. In each clause $C_i = y_1 \wedge \cdots \wedge y_l$, we have $y_j \in \{v_1, \dots, v_m\}$ and each variable can appear at most once. The #MDNF problem is to count the number of satisfying assignments of variables for the formula F.

We map the above instance of #MDNF to an uncertain graph ug, where each edge e_j corresponds to a variable v_j. Moreover, there are n possible world graphs, $pw_1(ug), \dots, pw_n(ug)$, from the uncertain graph ug, where $pw_i(ug)$ corresponds to a clause C_i.

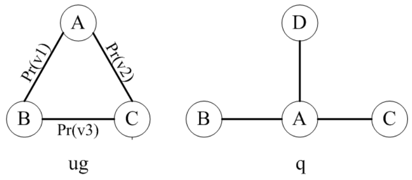


Figure 4: The uncertain graph ug and the query graph q constructed by $v_1 \vee v_2 \vee (v_1 \wedge v_2)$

For each possible world graph $pw_i(ug)$ and each edge e_j, we have $e_j \in pw_i(ug)$ if and only if v_j appears in the clause C_i. In other words, when a query graph q is given, a truth assignment satisfies F if and only if the corresponding possible world graph $pw_i(ug)$ is contained by q. Therefore, the probability $Pr(F)$ equals to the probability $Pr(ug \subseteq q)$. For instance, given a formula $F = v_1 \vee v_2 \vee (v_1 \wedge v_2)$, we map it to the corresponding uncertain graph ug in Figure 4. □

3. THE FILTERING-AND-VERIFICATION FRAMEWORK

In this section, our framework will be introduced as follows.

1. **Offline Index Construction**: Firstly, we discover all probabilistic frequent subgraphs as the initial candidate of the feature set. Then, we select the optimal feature set by the users' query logs. According to the feature set, we finally build a probabilistic inverted index, *PS-Index*.
2. **Filtering**: Based on the *PS-Index*, we test q against indexed features in feature set, F. Given the probabilistic threshold δ, if $f \nsubseteq q$ and $Pr(f \subset ug_i) \geq 1 - \delta$, ug_i can be safely pruned. Finally, we obtain the candidate of result set, $C_q = UGD - \cup_{f \in F} UGD_f(f \nsubseteq q \& Pr(f \subset ug_i) \geq 1 - \delta)$
3. **Verification**: For the uncertain graphs passing the filtering test, we first check whether each corresponding deterministic graph g_i of ug_i is a subgraph of the query graph q. If $g_i \subseteq q$, ug_i can be directly returned as the result. Otherwise, we use an unequal-probability-sampling-based approximation algorithm to verify the supergraph containment probability.

3.1 Cost Model Analysis

Given a query graph q, a feature set F, the query response time based on the above framework is:

$$\sum_{ug_i \in C_q} T_{SCP}(q, ug_i) + \sum_{f \in F} T_{SubIsm}(f, q) + T_{IndexSearch} \quad (3)$$

where $T_{SCP}(q, ug_i)$ is the time to check whether the supergraph containment probability between q and ug_i is greater than the given a probabilistic threshold $\delta (0 \leq \delta \leq 1)$, $T_{SubIsm}(f, q)$ is the time to test whether a feature f is the subgraph of the query graph q, (namely this time equals to the time of subgraph isomorphism test between the two deterministic graphs), and $T_{IndexSearch}$ is the time to find the candidate set C_q in the index.

According to the formula 3, we know that $T_{SCP}(q, ug_i)$ is a #P-hard operation, $T_{SubIsm}(f, q)$ is a NP-hard operation, and $T_{IndexSearch}$ can be finished in the polynomial time. Thus, it is reasonable that $T_{IndexSearch}$ is ignored in the total cost model. Moreover, the total cost can be simplified to the formula 4:

$$|C_q| \times AveT_{SCP} + |F| \times AveT_{SubIsm} \quad (4)$$

where $|C_q|$ is the size of the candidate set, $AveT_{SCP}$ is the average time of computing the supergraph containment probability, $|F|$ is the size of the feature set, $AveT_{SubIsm}$ is the average time of testing the subgraph isomorphism.

In addition, an important fact is that $AveT_{SCP}$ is usually several orders more than $AveT_{SubIsm}$, even though we adopt the approximation algorithm to compute SCP. This fact will be further verified in our experiments. In the deterministic environment, there is the subgraph isomorphism test instead of the computation of SCP. Thus, the formula 4 can be simplified to $(|C_q| + |F|) \times AveT_{SubIsm}$. To save the cost, existing works of supergraph query in deterministic graphs always try to minimize the size of $|C_q| + |F|$. However, in the problem of supergraph query in uncertain graphs, we have to first guarantee the minimum size of $|C_q|$, then try to decrease the size of $|F|$ as far as possible due to $AveT_{SCP} >> AveT_{SubIsm}$.

3.2 A Probabilistic Filtering Strategy

In Section 1, we introduce the basic idea of the probabilistic supergraph filtering strategy. Since the filtering strategy is the base of the feature selection and index construction, we provide the complete proof in this subsection. Moreover, we first define the concept of subgraph isomorphism probability due to the requirement of this filtering strategy.

DEFINITION 9. *(Subgraph Isomorphism Probability (SIP)) Given an uncertain graph ug and a deterministic query graph q, the subgraph isomorphism probability between q and ug is defined:*

$$Pr(q \subseteq ug) = \sum_{q \subseteq pw_i(ug) \cap pw_i(ug) \in PW(ug)} Pr(pw_i(ug)) \quad (5)$$

where $PW(ug)$ means the set all possible world graphs generated from ug, $q \subseteq pw_i(ug) \cap pw_i(ug) \in PW(ug)$ means the possible world graphs which include q.

Based on the definition of subgraph isomorphism probability, we have the following lemma.

LEMMA 1. *(The Upper Bound of Supergraph Containment Probability) Given a deterministic query graph q, an uncertain graph ug, and a feature subgraph f, if $f \nsubseteq q$ and $Pr(f \subseteq ug) = p$, then $Pr(ug \subseteq q) \leq 1 - p$.*

Proof (Sketch): According to the definition of SIP (Definition 9), we know that

$$Pr(f \subseteq ug) = \sum_{f \subseteq pw_i(ug) \cap pw_i(ug) \in PW(ug)} Pr(pw_i(ug))$$

and $f \nsubseteq q$. In addition, the above possible world graphs ($pw_i(ug) \in PW(ug)$ and $f \subseteq pw_i(ug)$) must not contain q. Thus, these possible world graphs which support probabilities to $Pr(f \subseteq ug)$ must not provide the probabilities to $Pr(ug \subseteq f)$. Hence, $Pr(ug \subseteq q) \leq 1 - p$ if $Pr(f \subseteq ug) = p$. Hence, the lemma holds. □

Based on Lemma 1, we can further obtain the probabilistic supergraph filtering strategy as follows.

THEOREM 2. *(Probabilistic Supergraph Pruning) Given a deterministic query graph q, an uncertain graph ug, a feature subgraph f, and a probabilistic threshold δ, if $f \nsubseteq q$ and $Pr(f \subseteq ug) > 1 - \delta$, then ug can be safely pruned.*

Proof (Sketch): Since $f \nsubseteq q$ and $Pr(f \subseteq ug) \geq 1 - \delta$, we can obtain $Pr(ug \subseteq q) < 1 - (1 - \delta) < \delta$ based on Lemma 1. Then, ug will not be in the final result set and can be safely pruned. □

4. PS-INDEX

In this section, we discuss in detail the *PS-Index* including the initial feature generation, feature selection and index construction.

4.1 Probabilistic Bounding-based Feature Selection

According to the probabilistic supergraph filtering strategy in Section 3.2, it is important for the pruning method to select significant features. Inspired by the previous researches of graph search in deterministic graphs [5, 21], we adopt a two-step approach to choose features. In the first step, we discover frequent subgraphs in uncertain graphs [26, 28] to produce the initial feature set, denoted F_0. In the second step, we select the final feature set, denoted F, from the initial feature set in order to remove redundant features.

In this subsection, we focus on how to generate the initial feature set efficiently. Although the existing works of uncertain graph search use the deterministic frequent subgraph mining technique to generate the initial feature set [23, 24], we will encounter two challenges in our problem if the deterministic frequent subgraph mining technique is directly adopted. 1) Which method should be used in both deterministic and uncertain frequent subgraph mining methods? 2) Could we enhance the efficiency of the feature generation process if we employ the uncertain frequent subgraph mining technique?

For the first challenge, the deterministic frequent subgraph mining methods ignore the uncertainty in uncertain graphs to generate the initial feature set including meaningless features. Since the existing solutions of uncertain subgraph query depend on the deterministic structural pruning, it needs the deterministic frequent subgraph mining methods to support. However, in the supergraph containment query, structural pruning does not work. Hence, we choose the uncertain frequent subgraph mining to generate the initial feature set for saving the uncertainty of uncertain graphs.

For the second challenge, the efficiency of uncertain frequent subgraph mining is the major bottleneck. Since the frequency of each subgraph pattern becomes a random variable in the uncertain environment, existing works of uncertain frequent subgraph mining include two different probabilistic semantics. One method uses the expectation of the frequency to measure whether each subgraph pattern is frequent [28]. The other method calculates the probability that the frequency is greater than *minsup* threshold to test whether this subgraph pattern is frequent [26]. We select the expectation-based framework since the recent research [19] proves that the expectation-based semantics is much faster and has the same effect with the probabilistic frequent semantics. Unfortunately, even though we employ the expectation-based semantics, the efficiency of uncertain frequent subgraph mining is still low due to the inherent high complexity of computing SIP. In order to speed up the process of feature generation, we propose a probabilistic-bounding-based method, which adopts a tight lower and upper bound to estimate the SIP of a subgraph pattern and an uncertain graph instead of the original exact check. In other words, we continue to use the existing expectation semantics-based uncertain frequent subgraph mining framework but replace the existing SIP computation method to the following strategies: 1) If the sum of lower bounds of an subgraph is greater than the threshold, the subgraph is frequent; 2) If the sum of upper bounds of an subgraph is lower than the threshold, the subgraph is infrequent; 3) If the threshold is in the interval between the sums of lower bounds and upper bounds, we have to adopt the existing sampling-based method to estimate the expectation of frequency of the subgraph. The two bounds are shown in the following lemma.

LEMMA 2. *(The Lower/Upper Bound of Subgraph Isomorphism Probability) Given a deterministic subgraph f, an uncertain graph ug (whose corresponding deterministic graph is denoted g), if there are m subgraph isomorphism instances of f in g, where SI_i represents the i-th subgraph isomorphism instance, the SIP of f and ug, $Pr(f \subseteq ug)$, satisfies,*

$$\begin{cases} Pr(f \subseteq ug) \geq \sum_{i=1}^{m} \frac{Pr(SI_i)^2}{Pr(SI_i)+\sum_{i\neq j} Pr(SI_i \cap SI_j)} \\ Pr(f \subseteq ug) \leq min\{\sum_{i=1}^{m} Pr(SI_i) - 2\sum_{i=2}^{m}\sum_{j=1}^{i-1} \frac{Pr(SI_i \cap SI_j)}{m}, 1\} \end{cases} \tag{6}$$

where $Pr(SI_i)$ means the existence probability of the i-th subgraph isomorphism instance.

Proof (Sketch): Because the SIP computation can be reduced the problem of computing a probabilistic DNF formula [24, 28], $Pr(f \subseteq ug)$ can be transformed to the form $Pr(SI_1 \vee \cdots \vee SI_m)$, where SI_i represents the i-th subgraph isomorphism instances. According to the *de Caen probability inequality* [8] and the *Kwerel probability inequality* [16], we can obtain the following lower upper and the upper bound of $Pr(SI_1 \vee \cdots \vee SI_m)$, respectively.

$$Pr(SI_1 \vee \cdots \vee SI_m) \geq \sum_{i=1}^{m} \frac{Pr(SI_i)^2}{Pr(SI_i) + \sum_{i\neq j} Pr(SI_i \cap SI_j)}$$

$$Pr(SI_1 \vee \cdots \vee SI_m) \leq \{\sum_{i=1}^{m} Pr(SI_i) - 2\sum_{i=2}^{m}\sum_{j=1}^{i-1} \frac{Pr(SI_i \cap SI_j)}{m}, 1\}$$

Thus, the lemma holds. □

According to Lemma 2, we can significantly enhance the efficiency of the feature generation process since the tight lower and upper bounds can be performed in polynomial time. Based on the initial feature set, we can employ the redundancy-aware feature selection algorithm [5] to select the final feature set consisting of an effective probabilistic inverted index, called *PS-Index*, to store these features in the next subsection.

4.2 Index Construction

After obtaining the selected feature set, we construct a prefix-search-tree-based index, which allocates a depth-first-search-based order for each feature in the index construction phase. The detail of *PS-Index* is defined as follows.

DEFINITION 10. *(PS-Index) Given an uncertain graph database $UGD = \{ug_1, \ldots, ug_n\}$ a initial feature set F_0 and a selected feature set $F = \{f_1, \ldots, f_t\}$, a PS-Index constructed on UGD consists of the following three components:*

A Feature Array. It stores the set of all features in F. Each element in this array has two pointers. A pointer is used to locate each feature in the prefix search tree. The other pointer points to the corresponding list of lower/upper bounds of each feature.

A List of Probabilistic Bound-based Arrays. It stores the set of all lower and upper bounds of selected features in each uncertain graph. Each unit in this array contain the lower and upper bounds of a feature in all uncertain graphs and is located by a pointer from the corresponding feature array.

A Prefix Search Tree. It is constructed in two steps. In the first step, the prefix search tree stores all features from F_0 , each node of such tree represents a feature by the DFS code, which translate a graph into a unique edge sequence, which is generated by performing a depth-first search (DFS) in a graph. [21]. Each node also has a pointer which can locate the corresponding element in feature array. In the second step, the nodes which is not chosen in F is labeled as an empty node. Please note that the second step can be quickly finished by the list of pointers from the feature array.

Based on the definition of *PS-Index*, the index can be constructed easily. Firstly, we can construct the prefix search tree by the depth-first-search-based order, meanwhile, the feature array is also built. Secondly, to the selected features, we build their probabilistic bound-based arrays in memory. For the features which fail to be selected, we set them as empty nodes in the prefix tree and store their probabilistic bound-based arrays in disk. Example 3 will further illustrate the construction of *PS-Index*.

EXAMPLE 3 (*PS-Index*). *Given an uncertain graph database which includes n uncertain graphs, a selected feature set F, we can build a PS-Index shown in Figure 5. The three components of the PS-Index is explained as follow. The feature array is in the most left in Figure 5. The list of probabilistic bound-based feature array is shown in the middle in Figure 5, where each row represents n pairs of lower and upper bounds that the corresponding feature graph is contained n uncertain graph. The prefix search tree is in the right in Figure 5. We can observe that each element of feature array has a pointer pointing the probabilistic bound-based feature*

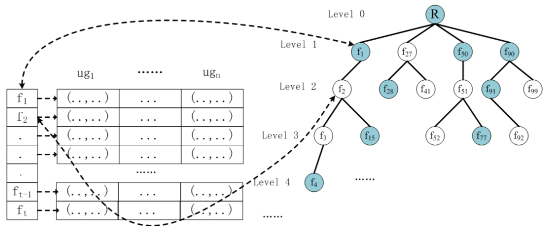


Figure 5: An Example of *PS-Index*

array and another pointer pointing such feature in the prefix tree (Due to the limited space, we only show two pair pointers of f_1 and f_2.). Meanwhile, each feature (it is actually stored via the DFS code [21]) has also a pointer to find the corresponding location in the feature array. For example, feature f_2 is loaded in the prefix tree, but it is set as empty node after loading the selected feature set. Thus, f_2 is shown as a white node. On the contrary, f_1 is a selected feature and is stored in the prefix search tree as a blue node in Figure 5.

Therefore, the *PS-Index* has two advantages. 1) In the query phase, the *PS-Index* has the anti-monotonic property of pruning. In the other words, if f_i is a prefix of f_j and f_i satisfies the pruning condition, f_j will not be checked. 2) The depth-first-search-order-based index structure can maximize the pruning power.

5. SAMPLING-BASED VERIFICATION

For the remaining uncertain graphs after the filtering phase, we have to check their supergraph containment probabilities (SCP). Because calculating *SCP* is known to be a #P-hard problem, we focus on how to design an efficient sampling-based approximation algorithm to solve it in this section. In Section 5.1, we review some backgrounds of sampling techniques and propose a simple-random-sampling-based algorithm to calculate *SCP*. In order to improve the efficiency of calculating *SCP*, we design an unequal-probability-sampling-based algorithm in Section 5.2.

5.1 Simple-Random-Sampling-based Approach

Since it is infeasible to calculate the SCP of the given query graph and an uncertain graph exactly due to the high computational complexity, the sampling-based solutions are practical and efficient. According to the sampling perspective, the population of this sampling is the set of all possible world graphs, the SCP is the estimated unknown proportion, which is also called the estimated parameter because sampling is essentially an estimation problem.

To facilitate our discussion, we define the concept of supergraph containment checking, which is an independent Bernoulli trial.

DEFINITION 11. *(supergraph containment checking) Given an uncertain graph ug, a possible world graph from ug, $pw_i(ug)$, and a deterministic query graph q, the supergraph containment checking between q and $pw_i(ug)$ is*

$$SCC(pw_i(ug)) = \begin{cases} 1 & pw_i(ug) \subseteq q \\ 0 & \textit{otherwise} \end{cases}$$

where $pw_i(ug) \subseteq q$ means that $pw_i(ug)$ is a subgraph of q.

Based upon the supergraph containment checking, we propose a *simple-random-sampling-based (SRS) algorithm* to calculate *SCP*, which is shown in Algorithm 1.

In the *SRS algorithm*, we sample n possible world graphs. For each possible world graph, we check whether the possible world graph ug_i is contained by the given deterministic query graph q. According to the supergraph containment checking, we can obtain the successful number of supergraph containment in n samples and

Algorithm 1: Simple-Random-Sampling-based Algorithm (SRS)

```
Input: an uncertain graph $ug$; a deterministic query graph $q$, the
       number of the sampling $n$.
Output: an estimated supergraph containment probability, $\widehat{SCP}$
1 $Z \leftarrow 0$;
2 for $i \leftarrow 1$ to $n$ do
3     Generate a possible world graph $pw_i(ug)$ from $ug$;
4     $Z \leftarrow Z + SCC(pw_i(ug))$;
5 $\widehat{SCP} \leftarrow \frac{Z}{n}$;
6 return $\widehat{SCP}$;
```

can calculate the mean as the estimated *SCP*. Moreover, Lemma 4 reports the bias and variance of *SRS algorithm*.

LEMMA 3. *(Unbiasedness and Variance of Simple-Random-Sampling Approach) The simple-random-sampling approach is unbiased. In addition, the variance of this approach is $\frac{1}{n}\widehat{SCP}(1-\widehat{SCP})$, where $\widehat{SCP}$ is the estimated SCP.*

Proof (Sketch): According to Algorithm 1, we know that the estimator (estimated *SCP*) is $\widehat{SCP} = \frac{1}{n}\sum_{i=1}^{n} SCC(pw_i(ug))$. Therefore, the expectation of $\widehat{SCP}$ is,

$$E[\widehat{SCP}] = E[\frac{1}{n}\sum_{i=1}^{n} SCC(pw_i(ug))] = \frac{1}{n}\sum_{i=1}^{n} E[SCC(pw_i(ug))] = Pr(ug \subseteq q)$$

Hence, the estimator $\widehat{SCP}$ is unbiased.

Since $SCC(pw_i(ug))$ is a random variable which is an independent Bernoulli trial, $\sum_{i=1}^{n} SCC(pw_i(ug))$ is a random variable following the Binomial distribution, the variance of $\widehat{SCP}$ is,

$$Var[\widehat{SCP}] = Var[\frac{1}{n}\sum_{i=1}^{n} SCC(pw_i(ug))] = \frac{1}{n^2}Var[\sum_{i=1}^{n} SCC(pw_i(ug))]$$
$$= \frac{1}{n}SCP(1-SCP) \approx \frac{1}{n}\widehat{SCP}(1-\widehat{SCP})$$

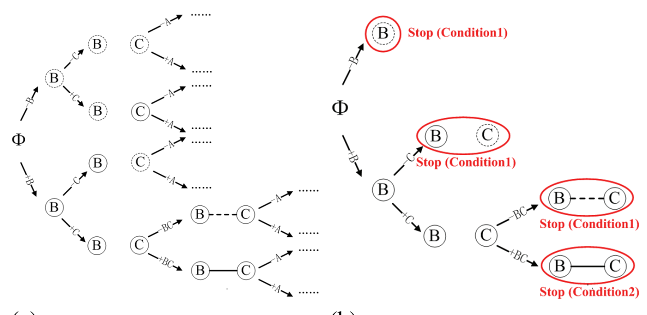
Thus, the lemma holds. □

5.2 Unequal-Probability-Sampling-based Approach

Although the simple-random-sampling-based approach can estimate *SCP*, it is not suitable for large-scale uncertain graphs and the high accurate requirement. The high accuracy leads to very large size of samplings, and each sampling needs to do a supergraph containment checking, which is a subgraph isomorphism test. Thus, the simple-random-sampling-based approach is infeasible for large uncertain graphs. Thus, we propose an efficient approximation algorithm, *unequal-probability-sampling-based (UPS) algorithm*, to estimate SCP. The basic idea is to group the space of samplings to avoid $2^{|V|+|E|}$ sampling spaces and reduce the number of subgraph isomorphism tests, where $|V|$ and $|E|$ are the size of vertices and edges of the corresponding deterministic graph of the given uncertain graph, respectively.

Based on our basic idea, we first introduce how to reduce the size of possible sampling spaces then discuss how to avoid the number of subgraph isomorphism tests. To facilitate our discussion, we introduce the concept of maximal common subgraph between a deterministic query graph and an uncertain graph.

DEFINITION 12. *(Maximum Common Subgraphs (MCS) [14]) Given an uncertain graph ug, a deterministic query graph q, the maximum common subgraphs of ug and q is the set of largest connected subgraphs of the corresponding deterministic graph of ug that is subgraph isomorphic to q, denoted as $MS = \{ms_1, \ldots, ms_m\}$. Note that the size of a graph is measured by the number of edges. The number of maximum common subgraphs may not be unique.*



(a) The Sampling Space of SRS algorithm in ug_2 (b) The Sampling Space of UPS algorithm in ug_2

Figure 6: The Sampling Spaces of SRS and UPG Algorithms

For example, given the uncertain graph ug_2 in Figure 1 and the query graph q in Figure 2, they have only one maximum common subgraph ms, which is the rightmost one in Figure 2.

Based on the concept of maximum common subgraphs, we propose an early stopping strategy, which can divide all possible world graphs into several groups and stop each sampling process as early as possible, in the following lemma. Since we only want to know which possible world graphs are contained or are not contained by the query graph, it is not necessary to sample each possible world graph exactly. Instead, we try to end each sample as early as possible as long as the containment relationship is known.

LEMMA 4. *(Early Stopping Strategy) Given an uncertain graph ug, a deterministic query graph q, the set of maximum common subgraphs $MS = \{ms_1, \ldots, ms_m\}$, and the current sampled graph g_s (the sets of vertices and edges of g_s are denoted as V_{in} or E_{in}, the sets of vertices and edges which are sampled and do not appear in g_s are denoted as V_{out} or E_{out}, respectively), this sampling can be stopped if the one of the following two conditions is satisfied,*

Condition 1: g_s must be contained by q if there is at least a maximum common subgraph ms_i in MS such that $V_{ug} \setminus V_{mc_i} \subseteq V_{out}$ and $E_{ug} \setminus E_{mc_i} \subseteq E_{out}$, where V_{ug}, V_{mc_i}, E_{ug} and E_{mc_i} are the sets of vertices and edges of ug and mc_i, respectively.

Condition 2: g_s must not be contained by q if $V_{in} \setminus V_q \neq \varnothing$ or $E_{in} \setminus E_q \neq \varnothing$, where V_q and E_q are the sets of vertices and edges of q, respectively.

Proof (Sketch): For the condition 1, $V_{ug} \setminus V_{mc_i}$ and $E_{ug} \setminus E_{mc_i}$ means the sets of vertices and edges which belong to the corresponding deterministic graph of ug and are not contained by q. If the condition holds, these aforementioned vertices and edges do not appear in the current sampled graph g_s, thus g_s must be contained by q.

For the condition 2, $V_{in} \setminus V_q \neq \varnothing$ or $E_{in} \setminus E_q \neq \varnothing$ means that there are at least one vertex or one edge which is in g_s but is not contained by q. Hence, g_s must not be contained by q under this condition. Thus, the lemma holds. □

We illustrate the early stopping strategy via the example.

EXAMPLE 4 (*Early Stopping Strategy*). *Given an uncertain graph ug_2 in Figure 1, a query graph q in the leftmost of Figure 2, and their maximum common subgraph in the rightmost of Figure 2, we show the complete sampling space of ug_2, which should include $2^{|V|+|E|}$ possible world graphs, in Figure 6(a). Each path in Figure 6(a) corresponds to a sampling in SRS algorithm. Figure 6(b) shows the sampling space based on the early stopping strategy. We can find that each sampling does not need to traverse a path instead of ends as early as possible. Thus, the early stopped sampled graph usually includes a set of possible world graphs. Note that a sampling process must end if this sampling only satisfies any early stopping condition. Hence, it is impossible that multiple early stopping conditions hold at the same time. To sum up, we can observe that the size of sampling space based on the early stopping strategy is significantly smaller than that of SRS algorithm.*

Algorithm 2: Unequal-Probability-Sampling-based Algorithm (UPS)

```
Input: an uncertain graph $ug$; a deterministic query graph $q$, the
       number of the sampling $n$.
Output: an estimated supergraph containment probability, $\widehat{SCP_{HT}}$
1 $MC \leftarrow$ Finding all maximal common subgraphs of $ug$ and $q$;
2 $\widehat{SCP_{HT}} \leftarrow 0$; $S \leftarrow 0$; $v \leftarrow 0$;
3 for $i \leftarrow 1$ to $n$ do
4     $s_i \leftarrow RS(ug, MC, q, n, \varnothing, \varnothing, \varnothing, \varnothing, 0)$;
5     if The i-th sampling is not a duplicate of previous sampling then
6         $S \leftarrow S + s_i$;
7         $v \leftarrow v + 1$;
8 return $\widehat{SCP_{HT}} \leftarrow \frac{S}{v}$;
```

Algorithm 3: RecursiveSampling (RS)

```
Input: an uncertain graph $ug$; a set of maximal common subgraphs
       $MC$, a deterministic query graph $q$, the number of the
       sampling $n$, the set of appearing vertices of $ug$ $V_{in}$, the set of
       appearing edges of $ug$ $E_{in}$, the set of disappearing vertices of
       $ug$ $V_{out}$, the set of disappearing edges of $ug$ $E_{out}$, the current
       probability $Pr$.
1 if $V_{in} \setminus V_q \neq \varnothing$ or $E_{in} \setminus E_q \neq \varnothing$ then
2     return 0; (Early Stopping Condition 2)
3 else if $\exists mc_i \in MC$ s.t. $V_{ug} \setminus V_{mc_i} \subseteq V_{out}$ and
  $E_{ug} \setminus E_{mc_i} \subseteq E_{out}$ then
4     return $\frac{Pr}{1-(1-Pr)^n}$; (Early Stopping Condition 1)
5 select a vertex or an edge, $ve$, from the remaining vertices and edges
  of $ug$;
6 if $ve$ is randomly decided to appear then
7     return RS($ug$,$MC$,$q$,$n$,$V_{in} \cup ve$(or
      $E_{in} \cup ve$),$V_{out}$,$E_{out}$,$p(ve)Pr$);
8 else
9     return RS($ug$,$MC$,$q$,$n$,$V_{in}$,$E_{in}$,$V_{out} \cup ve$(or
      $E_{out} \cup ve$),$(1-p(ve))Pr$);
```

The early stopping strategy reduces the size of the sampling space, we further hope to avoid the number of subgraph isomorphism tests as well. Thus, we employ a classical unbiased unequal-probability-sampling estimator, the Horvitz-Thompson estimator $\widehat{SCP_{HT}}$ [18], which can not only depend on the distinct samples but also provide a smaller variance than that of the simple-random-sampling estimator. In other words, the duplicate samples are not considered. Thus, the number of subgraph isomorphism tests is significant smaller than that of simple-random-sampling estimator, which has to test subgraph isomorphism for each sample.

DEFINITION 13. *(Horvitz-Thompson Estimator ($\widehat{SCP_{HT}}$)[18]) Given an uncertain graph ug, a query graph q, the size of samples n, the weight of each sample based on the early stopping strategy $w_i (1 \leq i \leq n)$, the Horvitz-Thompson estimator is*

$$\widehat{SCP_{HT}} = \sum_{i=1}^{v} \frac{w_i}{\pi_i}$$

where $\pi_i = 1 - (1 - q_i)^n$, q_i is the sampled probability, v is the number of distinct sampled graphs.

Based on the early stopping strategy and the Horvitz-Thompson estimator, we design an efficient unequal-probability-sampling-based (*UPS*) algorithm. The pseudo codes of *UPS* algorithm is shown in

Algorithm 2. This algorithm first finds the set of maximum common subgraphs in line 1. In lines 3-7, the algorithm performs n samples. In each sample, it recursively call a sub-procedure, called as *RecursiveSampling*, to calculate the parameters w_i and π_i of Horvitz-Thompson estimator. In addition, the sampled graph is not duplicate of previous sampling, it is just computed into the estimator. The final estimator, namely SCP, is returned in line 8.

In Algorithm 2, the core is the sub-procedure, *RecursiveSampling (RS)*, which is shown in Algorithm 3. The two conditions of the early stopping strategy are used to end the recursive procedure in lines 1-4. If the two conditions cannot be satisfied, this algorithm continues to select a new vertex or edge to sample randomly in lines 5-9. Note that this algorithm always prefers to choose vertices since the probabilities of edges are the conditional probabilities under two corresponding vertices already appear. It is easer to satisfy the early stopping strategy if we first sample vertices.

6. EXPERIMENTAL EVALUATION

In this section, we report the experimental results for the efficiency of proposed algorithms. In order to conduct a fair comparison, all the experiments are performed on an Intel(R) Core(TM) i7 3.40GHz PC with 4GB main memory, running on Microsoft Windows 7. Moreover, all the algorithms were implemented and compiled using Microsoft's Visual C++ 2010.

In order to test our proposed methods, we use a real-world uncertain graph database and a synthetic dataset which is a classical deterministic graph datasets with the synthetic probability distribution. For the real dataset, it comes from a real protein-protein interaction (PPI) network database, STRING[1], which includes the PPI networks of organisms. In a PPI network, vertices represent proteins, edges represent interactions between proteins, the labels of vertices are the COG functional annotations of proteins, and the existence probabilities of edges derive from the STRING database. We extract 1000 uncertain graphs from STRING. The average numbers vertices and edges of 1000 uncertain graphs are 43.2 and 97.5, respectively. Moreover, the average existence probabilities of edges in our dataset is 0.486. Please note that each numerical value in this dataset is always a 3-digit integer within the interval between 150 and 999. In fact, the integer is the decimal value of the corresponding probability. Besides the real uncertain graph database, we allocate probabilities which follow Gaussian distribution to a classical deterministic graph databset. Assigning probabilities to deterministic database in order to generate uncertain data is widely accepted by the previous related works [10, 24, 23]. The deterministic graph dataset, denoted AIDS [5], has been widely used to test deterministic graph queries. It extracts 10000 graphs from the original 43,905 structured chemical compounds. The average numbers vertices and edges of AIDS are 24.3 and 26.5, respectively. We set probabilities of vertices and edges with high mean (0.8) and low variance (0.1). The characteristics of and the default parameters above datasets are shown in Table 2. We also prepare the query sets and graph databases in a way similar to the existing uncertain graph queries [10, 24, 23]. Each query set q_i contains 100 connected query graphs. Note that i in q_i means the number of edges in q_i, such as q_50, q_100, etc. We also randomly select 2K, 4K, 6K, 8K and 10K graphs from AIDS and set probabilities to them for the scalability test. In addition, the sample size is set to be 1000 for SRS and UPS algorithms.

6.1 Time Efficiency Test

In this subsection, we verify the efficiency of the proposed algorithms and pruning strategies. We compare three algorithms:

[1]http://string-db.org/newstring_download/

Dataset	# of Graphs.	Ave. Vertices	Ave. Edge	Ave. Prob.
STRING	1000	43	97	0.486
AIDS	10000	24.3	26.5	0.8

Table 2: Characteristics and Default Parameters of Datasets

NoPruning, *SRSPruning*, and *UPSPruning*. *PS-Index*. *NoPruning* uses SRS algorithm in the verification phase and has on the pruning method (Theorem 2) in the filtering phase. Both *SRSPruning* and *UPSPruning* employ the *PS-Index* and pruning strategy but use SRS and UPS algorithms in their verification phase, respectively.

Varying Size of Query. Figures 7(a) and 7(b) show the total running time of the three competitive algorithms w.r.t. *size of query* in STRING and AIDS datasets, respectively. When the *size of query* increases, we observe that the running time of all the algorithms goes up. *UPSPruning* is always the fastest algorithm, *NoPruning* is the slowest one, and *SRSPruning* is faster than *NoPruning*.

It is reasonable because *UPSPruning* and *SRSPruning* employ the *PS-Index* and the probabilistic pruning to avoid most SCP computation. However, *NoPruning* has no the pruning and only calculates SCP with each uncertain graph. Furthermore, *UPSPruning* uses the early stopping strategy and the unequal-probability-sampling estimator to reduce the sampling spaces and avoid redundant subgraph isomorphism tests. Hence, *UPSPruning* outperforms *SRSPruning* in the two datasets.

Varying Probabilistic Threshold. Figures 7(c) - 7(d) report the running time w.r.t. *probabilistic threshold*. We can find that *UPSPruning* is the fastest algorithm in most of time. Different from the results w.r.t *size of query*, we observe that, by increasing *probabilistic threshold*, the total running time reduces. Moreover, the changing trends of the running time of *UPSPruning* and *SRSPruning* is relative stable. The pruning power of probabilistic pruning is further reported in the next subsection.

6.2 Pruning Power Test

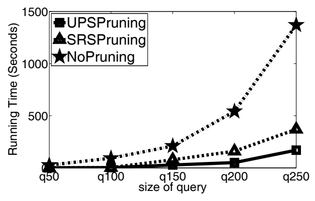
To better verify the effect of our proposed pruning, in this subsection, we report the pruning ratio of the probabilistic pruning on Figures 7(e) - 7(f). Due to the space limitation, we only show the pruning ratio w.r.t. *size of query* of the two sampling algorithms. The results of pruning ratio w.r.t other parameters are similar.

Varying Size of Query. Figures 7(e) and 7(f) show the pruning ratio w.r.t. *size of query* in STRING and AIDS datasets, respectively. The pruning ratio in STRING dataset is smaller than that on AIDS dataset. The smaller pruning ratio explains the reason that the computation saved in STRING is less than that in AIDS. Moreover, the less pruning ratio makes sense because the average probabilities of vertices and edges in AIDS follows the Gaussian distribution (mean=0.8, variance=0.1), most vertices and edges in AIDS likely appear. Thus, the probability of $Pr(f \subseteq ug)$ usually is high, the pruning condition in Theorem 2 can be satisfied easily.

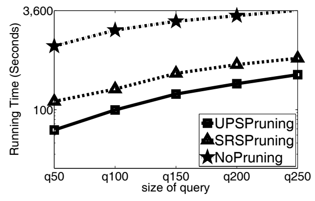
6.3 Verification Test

In this subsection, we report the verification time and the approximation quality of the two algorithms. However, due to the space limitation, we only show the verification time of the two sampling algorithms in STRING.

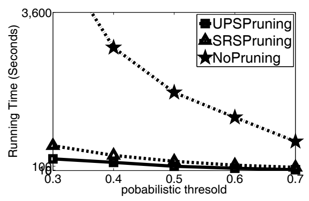
Varying Size of Query. Figures 7(g) shows the comparison of SRS and UPS algorithms in the verification step. We can observe that UPS algorithm outperforms SRS algorithm in efficiency. In particular, UPS is significantly faster than SRS when the *size of query* is large. This result also verifies that the early stopping strategy and unequal-probability-sampling-based estimator can reduce the sample space and avoid duplicate subgraph isomorphism tests.



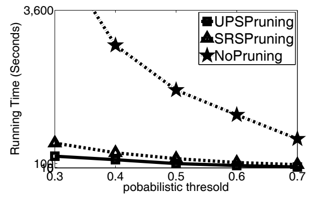
(a) STRING: time vs. size of query (δ=0.5)



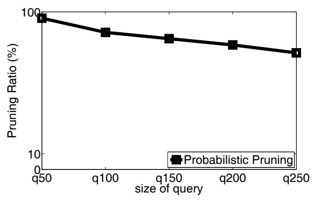
(b) AIDS: time vs. size of query (δ=0.5)



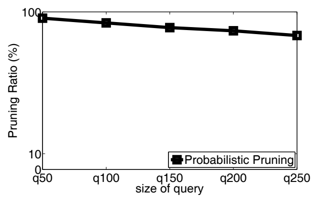
(c) STRING: time vs. threshold (size=150)



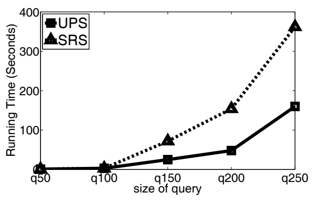
(d) AIDS: time vs. threshold (size=150)



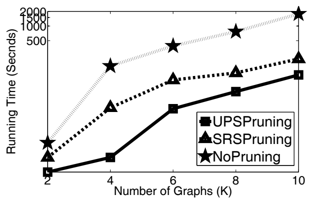
(e) STRING: pruning ratio vs. size of query (δ=0.5)



(f) AIDS: pruning ratio vs. size of query (δ=0.5)



(g) STRING: verification vs. size of query (δ=0.5)



(h) Scalability: time vs. size of databases

Figure 7: Performance of Efficiency, Pruning Power, Approximation Quality and Scalability

Table 3: Accuracy in Accident

Probabilistic Threshold	SRS Algorithm		UPS Algorithm	
	P	R	P	R
0.3	0.56	0.59	0.72	0.76
0.4	0.62	0.64	0.77	0.8
0.5	0.68	0.67	0.83	0.85
0.6	0.75	0.72	0.84	0.88
0.7	0.8	0.77	0.91	0.94

Approximation Quality. Besides offering efficient running time, the accuracy of sampling-based approximation algorithms is the other important evaluation criterion. We use the precision which equals $\frac{|AR \cap ER|}{|AR|}$ and the recall which equals $\frac{|AR \cap ER|}{|ER|}$ to measure the accuracy of approximation algorithms. Please note that AR means the result generated from the approximation algorithm, and ER is the result generated from the exact algorithm. Table 3 shows the precision and the recall w.r.t varying *probabilistic threshold* in the STRING dataset. That is because the variance of UPS estimator is usually lower than that of SRS.

6.4 Scalability Test

In this subsection, we report the scalability of our proposed algorithms. In Figure 7(h), increasing the number of uncertain graphs in the dataset from 2k to 10k, we observe that the running time curves of the $UPSPruning$ and $SRSPruning$ algorithms almost change linearly. However, the running time curve of *NoPruning* algorithm is exponential. In addition, the slopes of the curves of $UPSPruning$ and $SRSPruning$ are different. The slope of $UPSPruning$ is smaller than that of $SRSPruning$. This result is reasonable because $UPSPruning$ and $SRSPruning$ use the *PS-Index* and probabilistic pruning techniques. Most unqualified uncertain graphs are pruned in the filtering step. Furthermore, the early-stopping-strategy can reduced the running time in each sample, thus slope of curve of $UPSPruning$ is smaller.

7. RELATED WORK

In this section, we will review the related work from two categories, query processing in deterministic and uncertain graphs.

7.1 Deterministic Supergraph Queries

There are two types of closely related graph queries in deterministic graphs: subgraph query and supergraph containment query. Due to the limited space, we only discuss the related work of supergraph containment query in this subsection.

Supergraph containment query aims to find all graphs $g_i \in GD$ and $g_i \subseteq q$ from the given graph database GD. Most existing solutions adopt another filtering logic, called the exclusion logic [5], to avoid unnecessary subgraph isomorphism tests.

Firstly, Chen et al. [5] first proposed the concept of supergraph containment query and design a contrast-subgraph-based index (*c-index*). Especially, in order to obtain the maximum pruning power, *c-index* designs a redundancy-aware feature selection method. Zhang et al. [25] proposed a novel tree structure, *GPTree*, to integrate the graph database for saving the redundant subgraph isomorphism tests. In addition, a set of significant frequent subgraphs is mined as the indexing features in this method. Recently, Cheng et al. [6] design a new low-cost index, *IG-index*, based on a method of graph integration, which is used to approximately share the most common subgraphs in the graph database. In addition, based on this low-cost index, some result graphs can be directly returned without subgraph isomorphism test. Thus, the verification process will be dramatically speeded up. However, in uncertain scenario, the aforementioned approaches cannot be extended easily since edges in common subgraphs may have different probabilities. If we only store the lowest probability for each edge of a common subgraph, the pruning effect of the index will be reduced significantly.

7.2 Uncertain Graph Queries

In this subsection, we mainly review four kinds of uncertain graph query techniques. A probabilistic XML (*PXML*) database can be considered as a special uncertain graph database. Nierman et al. [12] first proposed a simple *PXML* model, *ProDB*, which used the probabilistic tree structure to model *PXML* databases. Kimelfeld et al. [11] summarized previous works of probabilistic XML and provided a series of efficient algorithms for different probabilistic XML models. Moreover, based on the $PrXML^{\{ind,mux\}}$ model, Chang et al. [4] proposed the efficient ranking algorithm for probabilistic twig matching results. The aforementioned studies mainly focused on the efficient probability computation over uncertain tree. Thus, they may have polynomial-time solution due to the constraints of the XML data. However, our work aims to a general uncertain graph query, which is #P-hard problem.

Besides querying over *PXML* databases, distance-based queries over uncertain graphs also attracted much attention recently. Yuan et al. [22] proposed the shortest path query over general uncertain graphs under the possible world semantics. Hua et al. [9] defined several probabilistic shortest path queries over uncertain road networks. In addition, Jin et al. [10] provided the work of reachability query over uncertain graph uncertain the possible world semantics. Potamias et al. [13] extended the concept of the uncertain distance and proposed several definition and efficient solutions of

kNN query over uncertain graphs. The above queries mainly base upon the concept of uncertain distances, which becomes a random variable in the uncertain graphs. However, our work handles multiple uncertain graphs rather than an uncertain graph.

Since PPI networks are considered as real uncertain graph datasets, the other related topic is to handle PPI networks data in bioinformatics. [7] aimed to predict protein functions based on different level neighbours and topological weights. Furthermore, [17] provided a comparison of different approaches estimating the protein interaction confidence, namely the probabilities of edges in uncertain graphs. Therefore, the aforementioned two works considered different issues with our paper.

In addition, the most closely related research to our work is about probabilistic subgraph query over uncertain graphs. Yuan et al. proposed the first work of probabilistic subgraph search over uncertain graphs [24]. Two efficient probabilistic pruning methods and a probabilistic inverted index were designed. Furthermore, Yuan et al. [23] defined the problem of probabilistic subgraph similarity search over uncertain graphs and provided the efficient solution, which included an effective matrix index and the lower and upper bound-based probabilistic pruning methods. Even though the two works utilizes the filtering-and-verification framework, our solution has the essential differences with the uncertain subgraph queries. Uncertain subgraph queries need to construct two indexes, deterministic graph index and probabilistic index, for the deterministic and probabilistic pruning, and aim to calculate the subgraph isomorphism probability. However, for the probabilistic supergraph containment query, the deterministic index has no contribution. Our work needs to efficiently handle the supergraph containment probability instead of the subgraph isomorphism probability. In addition, Zou et al. [27] proposed the problem of finding top-k maximal cliques in an uncertain graph. Besides the aforementioned work of uncertain graph queries, uncertain graph mining has also been studied recently. Zou et al. presented mining algorithm based on expected support semantic [28] and probabilistic semantic, respectively [26].

8. CONCLUSIONS

In this paper, a new uncertain graph query, probabilistic supergraph containment query, is proposed. We prove that this problem is #P-hard. Due to the high computational complexity, a probabilistic supergaph filtering strategy and a filtering-and-verification framework have been designed in order to avoid tedious computation. Moreover, a novel probabilistic inverted index, *PS-Index*, is developed to speed up the query processing. An unequal-probability-sampling-based algorithm is also proposed to efficiently compute supergraph containment probabilities in the verification phase. Extensive experiments on both real and synthetic datasets verify the effectiveness and efficiency of the proposed methods.

9. ACKNOWLEDGEMENTS

The authors thank Prof. Bogdan Cautis and the anonymous reviewers for their insightful and constructive comments. This work is supported in part by the Hong Kong RGC Project N_HKUST637/13, National Grand Fundamental Research 973 Program of China under Grant 2012-CB316200, National Natural Science Foundation of China (NSFC) Grant No. 61328202, Microsoft Research Asia Gift Grant, Microsoft Research Asia Fellowship 2012 and Google Faculty Award 2013.

10. REFERENCES

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

[2] S. Berretti, A. D. Bimbo, and E. Vicario. Efficient matching and indexing of graph models in content-based retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(10):1089–1105, 2001.

[3] K. R. Beutner, G. Prasad, E. Fletcher, C. DeCarli, and O. T. Carmichael. Estimating uncertainty in brain region delineations. In *IPMI*, pages 479–490, 2009.

[4] L. Chang, J. X. Yu, and L. Qin. Query ranking in probabilistic xml data. In *EDBT*, pages 156–167, 2009.

[5] C. Chen, X. Yan, P. S. Yu, J. Han, D.-Q. Zhang, and X. Gu. Towards graph containment search and indexing. In *VLDB*, pages 926–937, 2007.

[6] J. Cheng, Y. Ke, A. W.-C. Fu, and J. X. Yu. Fast graph query processing with a low-cost index. *VLDB J.*, 20(4):521–539, 2011.

[7] H. N. Chua, W.-K. Sung, and L. Wong. Exploiting indirect neighbours and topological weight to predict protein function from protein-protein interactions. *Bioinformatics*, 22(13):1623–1630, 2006.

[8] D. de Caen. A lower bound on the probability of a union. *Discrete Mathematics*, 169(1-3):217–220, 1997.

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

[10] R. Jin, L. Liu, B. Ding, and H. Wang. Distance-constraint reachability computation in uncertain graphs. *PVLDB*, 4(9):551–562, 2011.

[11] B. Kimelfeld, Y. Kosharovsky, and Y. Sagiv. Query efficiency in probabilistic xml models. In *SIGMOD Conference*, pages 701–714, 2008.

[12] A. Nierman and H. V. Jagadish. Protdb: Probabilistic data in xml. In *VLDB*, pages 646–657, 2002.

[13] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios. k-nearest neighbors in uncertain graphs. *PVLDB*, 3(1):997–1008, 2010.

[14] J. W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, 16(7):521–533, 2002.

[15] R. Saito, H. Suzuki, and Y. Hayashizaki. Interaction generality, a measurement to assess the reliability of a protein-protein interaction. *Nucleic Acids Research*, 30(5):1163–1168, 2002.

[16] Y. Sathe, M. Pradhan, and S. Shah. Inequalities for the probability of the occurrence of at least m out of n events. *Journal of Applied Probability*, pages 1127–1132, 1980.

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

[18] S. K. Thompson. *Sampling*. Wiley Desktop Editions, 2012.

[19] Y. Tong, L. Chen, Y. Cheng, and P. S. Yu. Mining frequent itemsets over uncertain databases. *PVLDB*, 5(11):1650–1661, 2012.

[20] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

[21] X. Yan, P. S. Yu, and J. Han. Graph indexing: A frequent structure-based approach. In *SIGMOD Conference*, pages 335–346, 2004.

[22] Y. Yuan, L. Chen, and G. Wang. Efficiently answering probability threshold-based shortest path queries over uncertain graphs. In *DASFAA (1)*, pages 155–170, 2010.

[23] Y. Yuan, G. Wang, L. Chen, and H. Wang. Efficient subgraph similarity search on large probabilistic graph databases. *PVLDB*, 5(9):800–811, 2012.

[24] Y. Yuan, G. Wang, H. Wang, and L. Chen. Efficient subgraph search over large uncertain graphs. *PVLDB*, 4(11):876–886, 2011.

[25] S. Zhang, J. Li, H. Gao, and Z. Zou. A novel approach for efficient supergraph query processing on graph databases. In *EDBT*, pages 204–215, 2009.

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

[27] Z. Zou, J. Li, H. Gao, and S. Zhang. Finding top-k maximal cliques in an uncertain graph. In *ICDE*, pages 649–652, 2010.

[28] Z. Zou, J. Li, H. Gao, and S. Zhang. Mining frequent subgraph patterns from uncertain graph data. *IEEE Trans. Knowl. Data Eng.*, 22(9):1203–1218, 2010.