Discovering Frequent Subgraphs over Uncertain Graph Databases under Probabilistic Semantics

Zhaonian Zou
znzou@hit.edu.cn

Hong Gao
honggao@hit.edu.cn

Jianzhong Li
lijzh@hit.edu.cn

School of Computer Science and Technology
Harbin Institute of Technology
92 West Da Zhi Street, Harbin, Heilongjiang 150001, China

ABSTRACT

Frequent subgraph mining has been extensively studied on certain graph data. However, uncertainties are inherently accompanied with graph data in practice, and there is very few work on mining uncertain graph data. This paper investigates frequent subgraph mining on uncertain graphs under probabilistic semantics. Specifically, a measure called φ-frequent probability is introduced to evaluate the degree of recurrence of subgraphs. Given a set of uncertain graphs and two numbers $0 < \varphi, \tau < 1$, the goal is to quickly find all subgraphs with φ-frequent probability at least τ. Due to the NP-hardness of the problem, an approximate mining algorithm is proposed for this problem. Let $0 < \delta < 1$ be a parameter. The algorithm guarantees to find any frequent subgraph S with probability at least $\left(\frac{1-\delta}{2}\right)^s$, where s is the number of edges of S. In addition, it is thoroughly discussed how to set δ to guarantee the overall approximation quality of the algorithm. The extensive experiments on real uncertain graph data verify that the algorithm is efficient and that the mining results have very high quality.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: [data mining]

General Terms

Algorithms, Performance, Theory

Keywords

Uncertain graph, frequent subgraph, probabilistic semantics

1. INTRODUCTION

Graphs are general data structures for representing complicated relationships between objects, which have seen wide applications in bioinformatics, social networks and spatial databases, etc. Large amount of data represented by graphs, also known as *graph data*, call for intelligent tools to analyze and understand them. *Frequent subgraph mining* [9] is one of the powerful tools to study the structures of graph data, especially, the recurring substructures. More precisely, the mining task is formulated as follows. Given a set D of graphs and a number $0 < \varphi < 1$, find all subgraphs that occur in at least $\varphi \cdot |D|$ graphs in D. The percentage of graphs in D that contain a subgraph S is called the *support* of S.

Recent researches [4,13,14] have shown that *uncertainties* are inherent in graph data, in particular, the structures of graphs are uncertain. In the data uncertainty models used in [4, 13], each edge of a graph is associated with a number indicating the probability of the edge existing in reality, and the existence of edges is mutually independent. Such graph is called an *uncertain graph* [13]. An uncertain graph essentially represents a probability distribution over all of the certain graphs in the forms of which the uncertain graph may actually exist. Each of these certain graphs is called an *implicated graph* [13].

Zou et al. [13] studied frequent subgraph mining on uncertain graph data. A set of uncertain graphs $D = \{G_1, G_2, \ldots, G_n\}$ represents a probability distribution over a family $\mathcal{F}$ of certain graph sets. Each set of certain graphs $D' = \{G'_1, G'_2, \ldots, G'_n\} \in \mathcal{F}$ satisfies that $G'_i \in D'$ is an implicated graph of $G_i \in D$ for $1 \leq i \leq n$. For a certain subgraph S, the degree of recurrence of S in D is measured by the expected value of the supports of S in all certain graph sets in $\mathcal{F}$, called the *expected support* of S. Thus, the frequent subgraph mining problem on uncertain graph data is defined in [13] as follows. Given a set D of uncertain graphs and a number $0 < \gamma < 1$, find all subgraphs with expected support at least γ.

Motivated by some recent work [2,11] on frequent itemset mining on probabilistic data, this paper investigates frequent subgraph mining on uncertain graph data based on semantics different from that adopted by [13]. Again, a set D of uncertain graphs represents a probability distribution over a family $\mathcal{F}$ of certain graph sets. However, the degree of recurrence of a certain subgraph S in D is measured by the probability that S has support at least φ across all certain graph sets in $\mathcal{F}$. Such probability is called *φ-frequent probability* in this paper. Therefore, the problem to be solved in this paper can be defined as follows. Given a set D of uncertain graphs and two numbers $0 < \varphi, \tau < 1$, find all subgraphs with φ-frequent probability at least τ.

Though the difference between the two semantics for mining uncertain data has been argued in [11] from the aspect of mining results, there is a lack of explanation of the difference from the angle of application areas. In short, frequent

KDD'10, July 25–28, 2010, Washington, DC, USA.

subgraph mining under expected semantics [13] is more suitable for investigating structural patterns in uncertain graph data, which behaves more like an exploratory tool; whereas, frequent subgraph mining under probabilistic semantics in this paper is more suitable for extracting features from uncertain graph data. Supposing a subgraph having support at least φ can be a feature in a set of certain graphs, then φ-frequent probability perfectly captures the probability of a subgraph being a feature in a set of uncertain graphs, but expected support does not. On the contrary, expected support characterizes the expected degree of recurrence of a subgraph, but φ-frequent probability does not.

Although the problem to be solved in this paper follows the same semantics as the frequent itemset mining problems in [2, 11], the algorithms presented in [2, 11] can not be extended to our problem. This is because it can be decided in polynomial time whether an itemset is frequent or not [2,11]; however, it is #P-*hard* to compute the φ-frequent probability of a subgraph as will be proved later in this paper.

The problem is also proved to be a NP-hard problem. Hence, instead of discovering all strictly frequent subgraphs, we find an ε-approximate set of frequent subgraphs including all frequent subgraphs and a fraction of infrequent subgraphs but with φ-frequent probability at least $\tau-\varepsilon$, where $0 < \varepsilon < \tau$ is a small error tolerance. In other words, all subgraphs with φ-frequent probability at least τ must be output, but all subgraphs with φ-frequent probability less than $\tau - \varepsilon$ need not to be output.

This paper proposes an approximate mining algorithm to find such an ε-approximate set of frequent subgraphs. Let $0 < \delta < 1$ be a pre-specified parameter. The algorithm guarantees to discover any frequent subgraph S with probability at least $\left(\frac{1-\delta}{2}\right)^s$, where s is the number of edges of S.

The algorithm is developed based on the well-known *gSpan* algorithm [9]. First, all subgraphs are encoded into their *minimum DFS codes* and are organized into a search tree according to the *lexicographic order* of minimum DFS codes [9]. Then, the search tree is traversed in depth-first order to find an ε-approximate set of frequent subgraphs. In particular, for each visited subgraph S, instead of computing the exact φ-frequent probability of S, it is determined whether the φ-frequent probability of S is *certainly* not less than $\tau - \varepsilon$ and is *probably* greater than or equal to τ using an approximation algorithm. The approximation algorithm is very simple and fails with probability at most δ. If the answer from the approximation algorithm is "yes", then output S and continue the search, otherwise stop searching the descendants of S since the φ-frequent probability of any supergraph of S is definitely no greater than τ.

The theoretical analysis shows that to obtain any frequent subgraph with probability at least $1-\Delta$, δ should be at most $1 - 2 \cdot (1 - \Delta)^{1/\ell_{\max}}$, where $\ell_{\max}$ is the maximum number of edges of frequent subgraphs. In addition, extensive experiments were performed on real uncertain graph data to evaluate the practical performance and the approximation quality of the proposed algorithm. The experimental results verify that the algorithm is very efficient and accurate.

The main contributions of this paper are as follows.

- This paper formally defines the problem of mining frequent subgraphs on uncertain graph data under probabilistic semantics.
- It is rigorously proved that the mining problem is NP-hard and that it is #P-hard to compute the φ-frequent probability of a subgraph.
- An approximate mining algorithm is proposed to find an ε-approximate set of frequent subgraphs. The algorithm guarantees to discover any frequent subgraph S with probability at least $\left(\frac{1-\delta}{2}\right)^s$, where s is the number of edges of S.
- It is thoroughly discussed how to set parameter δ to guarantee the overall approximation quality of the proposed algorithm.
- Extensive experiments were carried out on real uncertain graph data to evaluate the performance and the approximation quality of the proposed algorithm.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the model of uncertain graph data and defines the frequent subgraph mining problem. Section 4 formally proves the inherent computational complexity of the problem. Section 5 proposes the mining algorithm and analyzes the performance guarantees of the algorithm. Section 6 points out how to set parameters to ensure the overall approximation quality of the algorithm. Section 7 shows the experimental results on real uncertain graph data. Finally, Section 8 concludes this paper.

2. RELATED WORK

A number of algorithms have been proposed for mining frequent subgraphs on certain graph data, which have been surveyed in [1]. All these algorithms can not handle uncertainties inherent in graph data. However, some state-of-the-art techniques adopted by the algorithms such as *minimum DFS codes* [9] for representing subgraphs and the *right-most extension* technique [9] for extending subgraphs can also be used in our work because knowledge to be discovered in this paper is actually certain subgraphs.

Kimmig and Raedt [6] cast pattern mining problems in the context of *logic programming*, particularly in *ProbLog*, a probabilistic Prolog system. Due to the powerful expression capability, ProbLog can represent uncertainties of itemsets, sequences, trees or graphs. Their appealing method is an integration of *multi-relational data mining* and *inductive logic programming*. However, due to the data representation in ProbLog, operations on graphs such as subgraph isomorphism testings are implemented by *clause reductions*, which become inefficient on large-sized graphs.

Zou et al. [13] studied frequent subgraph mining on uncertain graph data independently. They proposed *expected support* to evaluate the significance of subgraphs. In particular, the expected support of a subgraph S is the expected value of the supports of S in all implicated graph databases. It differs from φ-frequent probability proposed in this paper in the following way. Even if the support of S can be less than φ in a specific implicated graph database, it still contributes to the expected support of S but doesn't contribute to the φ-frequent probability of S.

Although the problem in this paper follows the same semantics as the frequent itemset mining problems in [2, 11], the algorithms proposed in [2,11] can not be extended to our problem. Except for the difference in data type, another important reason is that it can be decided in polynomial time whether an itemset is frequent or not; however, it is #P-hard to compute the φ-frequent probability of a subgraph.

3. PROBLEM DEFINITION

3.1 Uncertain Graphs

Let us first extend the model of *uncertain graphs* recently proposed in [13] to the following one that considers uncertainties of both vertices and edges.

Definition 1. An *uncertain graph* is a system $G = (V, E, \Sigma, L_V, L_E, P_V, P_E)$, where (V, E) is an undirected graph, V is the set of vertices, E is the set of edges, Σ is a set of *labels*, $L_V : V \to \Sigma$ is a function assigning labels to the vertices, $L_E : E \to \Sigma$ is a function assigning labels to the edges, $P_V : V \to [0, 1]$ is a function assigning *existence probability* values to the vertices, and $P_E : E \to [0, 1]$ is a function assigning *conditional existence probability* values to the edges given their endpoints.

The existence probability, $P_V(v)$, of a vertex $v \in V$ is the probability of v existing in practice. The conditional existence probability, $P_E(e|u, v)$, of an edge $e = (u, v) \in E$ is the probability of e existing between vertices u and v on the condition that both u and v exist in practice. Thus, a labeled graph in traditional graph mining [9], which is called *certain graph* in this paper, is a special uncertain graph with existence probability values of 1 on all vertices and conditional existence probability values of 1 on all edges.

Similar to the statement in [13], an uncertain graph essentially represents a set of certain graphs *implicated* by it, each of which is a possible structure in the form of which the uncertain graph might be present in practice.

Definition 2. An uncertain graph $G = (V, E, \Sigma, L_V, L_E, P_V, P_E)$ *implicates* a certain graph $G' = (V', E', \Sigma', L'_V, L'_E)$, denoted by $G \Rightarrow G'$, if $V' \subseteq V$, $E' \subseteq E \cap (V' \times V')$, $\Sigma' \subseteq \Sigma$, $L'_V = L_V|_{V'}$, and $L'_E = L_E|_{E'}$, where $E \cap (V' \times V')$ is the set of edges with both endpoints in V', and $L|_X$ denotes the function obtained by restricting function L to domain X.

This paper assumes that the existence probabilities of vertices and the conditional existence probabilities of edges of an uncertain graph are *mutually independent*, respectively. Based on this assumption, the probability of an uncertain graph $G = (V, E, \Sigma, L_V, L_E, P_V, P_E)$ implicating a certain graph $G' = (V', E', \Sigma', L'_V, L'_E)$ is

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

where $E \cap (V' \times V')$ is the set of edges with both endpoints in V'.

Let $Imp(G)$ denote the set of all implicated graphs of an uncertain graph G. It is easy to verify that

$$|Imp(G)| = O\left(\sum_{i=1}^{|V|} \binom{|V|}{i} 2^{i(i-1)/2} \right)$$

by counting the number of implicated graphs of a fully-connected uncertain graph. Due to arguments similar to the proof of Theorem 1 in [15], we have the following theorem.

THEOREM 1. *For an uncertain graph G, the function given in Equation 1 defines a probability distribution over $Imp(G)$.*

3.2 Uncertain Graph Databases

Let us proceed to extend the model of *uncertain graph databases* proposed in [13] to the following one that takes *trivial* implicated graphs into account.

An *uncertain graph database* is a set of uncertain graphs. It essentially represents a set of *implicated graph databases*.

Definition 3. An uncertain graph database $D = \{G_1, G_2, \ldots, G_n\}$ *implicates* a certain graph database $D' = \{G'_1, G'_2, \ldots, G'_m\}$ if $m \leq n$, and there is an injection $\sigma : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, n\}$ such that $G_{\sigma(i)} \Rightarrow G'_i$ for $1 \leq i \leq m$.

Note that, in Definition 3, we have $m \leq n$ because every uncertain graph in D generally has a non-zero probability to implicate a *trivial* certain graph $\emptyset$, i.e., a certain graph with no vertices and no edges. Since trivial graphs do not contain useful knowledge, they should be eliminated from implicated graph databases. This is the main difference from the model of uncertain graph databases proposed in [13].

This paper assumes that the uncertain graphs in an uncertain graph database are *mutually independent*. Based on this assumption, the probability of an uncertain graph database $D = \{G_1, G_2, \ldots, G_n\}$ implicating a certain graph database $D' = \{G'_1, G'_2, \ldots, G'_m\}$ is

$$\Pr(D \Rightarrow D') = \left(\prod_{i=1}^{m} \Pr(G_{\sigma(i)} \Rightarrow G'_i) \right) \cdot \left(\prod_{i \in \{1,2,\ldots,n\} \setminus \{\sigma(x) | 1 \leq x \leq m\}} \Pr(G_i \Rightarrow \emptyset) \right), \quad (2)$$

where σ is an injection from $\{1, 2, \ldots, m\}$ to $\{1, 2, \ldots, n\}$ such that $G_{\sigma(i)} \Rightarrow G'_i$ for $1 \leq i \leq m$, and $\emptyset$ denotes a trivial graph.

Let $Imp(D)$ denote the set of all implicated graph databases of an uncertain graph database D. Based on the independence assumption, it is easy to know that $|Imp(D)| = \prod_{G \in D} |Imp(G)|$. Using the arguments in the proof of Theorem 2 in [15], we can prove the theorem below.

THEOREM 2. *For an uncertain graph database D, the function given in Equation 2 defines a probability distribution over $Imp(D)$.*

3.3 Frequent Subgraph Mining

Definition 4. A certain graph $G = (V, E, \Sigma, L_V, L_E)$ is *subgraph isomorphic* to another certain graph $G' = (V', E', \Sigma', L'_V, L'_E)$, denoted by $G \sqsubseteq G'$, if there exists an injection $f : V \to V'$ such that

(1) $L_V(v) = L'_V(f(v))$ for any $v \in V$,

(2) $(f(u), f(v)) \in E'$ for any $(u, v) \in E$,

(3) $L_E((u, v)) = L'_E((f(u), f(v)))$ for any $(u, v) \in E$.

Injection f is called a *subgraph isomorphism* from G to G'.

Probabilistic frequent subgraphs. As in traditional frequent subgraph mining, this paper also considers mining

connected subgraphs. If not otherwise specified, a subgraph refers to a connected subgraph in the rest of the paper.

Let D be an uncertain graph database, $Imp(D)$ be the set of all implicated graph databases of D, and S be a certain subgraph. Because each implicated graph database $D' \in Imp(D)$ is actually a certain graph database, the traditional definition of *support* [7] applies to D', that is, the support of S in D' is

$$sup(S; D') = \frac{|\{G' \in D' | \, S \sqsubseteq G'\}|}{|D'|}. \quad (3)$$

Thus, the probability that the support of S is no less than $0 < \varphi < 1$ across all implicated graph databases of D is

$$\Pr(S; D, \varphi) = \sum_{D' \in Imp(D),\, sup(S;D') \geq \varphi} \Pr(D \Rightarrow D'), \quad (4)$$

where $\Pr(D \Rightarrow D')$ is the probability of D implicating D' as given in Equation 2. In the rest of the paper, the probability given in Equation 4 is called the *φ-frequent probability* of S in D. When D and φ are explicit from the context, $\Pr(S; D, \varphi)$ can be simply written as $\Pr(S)$. A subgraph S is called *(φ, τ)-probabilistic frequent* if the φ-frequent probability of S is no less than a user-specified confidence threshold $0 < \tau < 1$. When φ and τ are clear from the context, S can be simply called a frequent subgraph.

Problem statement. The problem of mining frequent subgraphs in an uncertain graph database under probabilistic semantics can be formalized as follows.

Input: an uncertain graph database D, a support threshold $0 < \varphi < 1$, and a confidence threshold $0 < \tau < 1$.

Output: all (φ, τ)-probabilistic frequent subgraphs in D.

4. INHERENT COMPUTATIONAL COMPLEXITY OF THE PROBLEM

This section formally proves the inherent computational complexity of the problem of mining frequent subgraphs in an uncertain graph database. The following proofs use the complexity class #P for enumeration problems [8].

THEOREM 3. *It is #P-hard to compute the φ-frequent probability of a subgraph S in an uncertain graph database D.*

PROOF. We prove the theorem by reducing an arbitrary instance of the #P-*complete* problem of counting the number of assignments satisfying a monotone k-DNF formula F [8] to an instance of the current problem in polynomial time. Here, let $F = C_1 \vee C_2 \vee \cdots \vee C_m$ contain m clauses over n variables $x_1, x_2, \ldots, x_n$. Each clause C_i is in the form of $l_1 \wedge l_2 \wedge \cdots \wedge l_k$, where k is a constant, and each literal l_j is an un-negated variable in $\{x_1, x_2, \ldots, x_n\}$. Without loss of generality, we assume that a variable occurs in a clause at most once. The reduction is carried out as follows.

Firstly, construct an uncertain graph database $D = \{G\}$, where G is a bipartite uncertain graph. The vertex set of G is $U \cup V$, where $U = \{c_1, c_2, \ldots, c_m\}$ and $V = \{v_1, v_2, \ldots, v_n\}$. All of the vertices in U are labeled by α and have existence probabilities of 1, and all of the vertices in V are labeled by β and have existence probabilities of 1/2. There is an edge between vertices c_i and v_j if and only if clause C_i contains variable x_j. All of the edges of G are labeled by γ and have conditional existence probabilities of 1.

Secondly, create a certain subgraph S. The vertex set of S is $\{c, v_1, v_2, \ldots, v_n\}$, where c is labeled by α, and $v_1, v_2, \ldots, v_n$ are labeled by β. There is an edge labeled by γ connecting vertex c with each of $v_1, v_2, \ldots, v_n$.

Thirdly, let $\varphi = 1$.

The correspondence between the number of assignments satisfying F and the φ-frequent probability, $\Pr(S; D, \varphi)$, of S in D can be established due to the following arguments.

(1) A truth assignment π 1-to-1 corresponds to an implicated graph database D'_π of D such that variable x_i is assigned true in π if and only if vertex v_i is contained in the only certain graph in D'_π.

(2) A truth assignment π satisfies F if and only if S has support at least φ in the implicated graph database D'_π that π 1-to-1 corresponds to.

Subsequently, the number of truth assignments satisfying F is $2^n \cdot \Pr(S; D, \varphi)$. The reduction is completed. □

THEOREM 4. *It is #P-hard to count the number of frequent subgraphs in an uncertain graph database.*

PROOF. An instance of the #P-hard problem of counting the number of frequent subgraphs with support at least φ in a certain graph database D [10] can be trivially reduced to an instance of the current problem by specifying the input uncertain graph database to be D, the support threshold to be φ and the confidence threshold $\tau = 1$. This is because D is a special uncertain graph database, and a subgraph has support at least φ in D if and only if its φ-frequent probability in D is 1. Thus, the theorem holds. □

5. APPROXIMATE MINING ALGORITHM

Due to the hardness results proved in Section 4, it can not be expected to discover all strictly frequent subgraphs in polynomial time unless P = NP. Instead, an approximate mining algorithm is proposed to find a broader set of subgraphs including all frequent subgraphs and a fraction of infrequent subgraphs but with φ-frequent probability at least $\tau - \varepsilon$, where $0 < \varepsilon < \tau$ is an error tolerance. In other words, all subgraphs with φ-frequent probability at least τ must be output, but all subgraphs with φ-frequent probability less than $\tau - \varepsilon$ need not to be output.

It is not meant to discover all subgraphs with φ-frequent probability at least $\tau - \varepsilon$; or otherwise, the problem would become even harder. Suppose a subgraph S has φ-frequent probability at least $\tau - \varepsilon$. If it is definitely known that the φ-frequent probability of S is less than τ, then S need not to appear in the mining results.

5.1 High-Level Description

The approximate mining algorithm takes as input an uncertain graph database D, a support threshold φ, a confidence threshold τ, and an error tolerance ε. The procedure of the algorithm can be briefly outlined as follows.

(S1) First, organize all subgraphs in D into a *search tree*, where nodes represent subgraphs, and each node (subgraph) is subgraph isomorphic to all its children, if it has any, and has one less edge than all of them.

(S2) Then, examine the subgraphs in the search tree in depth-first order. For each examined subgraph S, determine in polynomial time whether S has φ-frequent

probability at least $\tau-\varepsilon$ and probably at least τ. If the answer is "yes", then output S and proceed to examine the descendants of S in depth-first order. Otherwise, all descendants of S are certainly infrequent and can be pruned due to the following property of φ-frequent probability.

LEMMA 1 (APRIORI PROPERTY). *For two subgraphs S and S', if $S \sqsubseteq S'$, then $\Pr(S) \geq \Pr(S')$.*

PROOF. The lemma can be readily proved from Equation 4 and the Apriori property of support [7]. □

The algorithm terminates while no subgraphs are left unexamined. Next, we clarify the details of each step.

Building search tree of subgraphs. Because subgraphs are actually certain graphs, all subgraphs in D can be encoded into *minimum DFS codes*, the state-of-the-art canonical graph coding scheme developed for frequent subgraph mining [9]. Informally speaking, the minimum DFS code of a subgraph is a sequence that is prior to all other sequential representations of the isomorphic subgraphs according to the *lexicographic order* of DFS codes. For more details on minimum DFS codes, please refer to [9].

Then, the search tree of subgraphs can be constructed. The nodes are all subgraphs in D. The root is the trivial subgraph $\emptyset$. The parent $parent(S)$ of each subgraph S ($S \neq \emptyset$) satisfies that the minimum DFS code of $parent(S)$ is the longest prefix of that of S.

Depth-first search on search tree. Note that we do not materialize the search tree in memory but enumerate subgraphs in the search tree in depth-first order. Particularly, for each subgraph S, all its children are generated as follows. First, perform *right-most extension* [9] to S, obtaining a set of subgraphs, each of which contains S and has one more edge than S. Then, for each right-most extended subgraph S', if the minimum DFS code of S is a prefix of that of S', then S' is a child of S, otherwise S' is not a child of S. Please refer to [9] for more details on right-most extension.

Verifying visited subgraphs. For each subgraph S visited in the search, it need to be verified whether S can be output as a result or not using the following method. First, approximate the φ-frequent probability, $\Pr(S)$, of S by an interval $[p_l, p_u]$ such that $\Pr(S) \in [p_l, p_u]$ and that $|p_u - p_l| \leq \varepsilon$. Then, apply the following rules.

(R1) If $p_l \geq \tau-\varepsilon$ and $p_u \geq \tau$, then it is certain that $\Pr(S) \geq \tau - \varepsilon$ and is probable that $\Pr(S) \geq \tau$, thus S can be output as a result.

(R2) If $p_u < \tau$, then $\Pr(S) < \tau$, thus S is not frequent.

Section 5.2 presents the details of the algorithm for approximating $\Pr(S)$ by such interval $[p_l, p_u]$.

5.2 Approximating φ-Frequent Probability

This subsection proposes an algorithm for approximating the φ-frequent probability, $\Pr(S)$, of a subgraph S by an interval $[p_l, p_u]$ such that $|p_u - p_l| \leq \varepsilon$ and $\Pr(S) \in [p_l, p_u]$. Let $0 < \delta < 1$ be a parameter. The algorithm fails to produce such an interval with probability at most δ. This subsection first develops a dynamic programming-based method to exactly compute $\Pr(S)$ in exponential time and then extends it to produce desired interval $[p_l, p_u]$ with probability at least $1 - \delta$ in polynomial time.

5.2.1 Dynamic Programming

Let us begin with a concept originally introduced in [13]. We say "a subgraph S occurs in an uncertain graph G" if S is subgraph isomorphic to at least one implicated graph of G. Naturally, the probability of S occurring in G is

$$\Pr(S \sqsubseteq_U G) = \sum_{G' \in Imp(G),\, S' \sqsubseteq G'} \Pr(G \Rightarrow G'), \qquad (5)$$

where $Imp(G)$ is the set of all implicated graphs of G, and $\Pr(G \Rightarrow G')$ is the probability of G implicating G' as given in Equation 1.

With the concept given above, the φ-frequent probability of a subgraph S can be exactly computed via dynamic programming. Suppose the uncertain graphs in D are indexed from 1 to n, i.e., $D = \{G_1, G_2, \ldots, G_n\}$. Let $T[0..n, 0..n, 0..n]$ be a three-dimensional array with subscript ranging from 0 to n in each dimension. Each element $T[i, j, k]$ of T stores the probability that, across all implicated graph databases of $\{G_1, G_2, \ldots, G_k\} \subseteq D$,

(1) the number of certain graphs in the implicated graph database is $i + j$, and

(2) subgraph S is subgraph isomorphic to i certain graphs and is not subgraph isomorphic to j certain graphs in this implicated graph database.

All elements of T can be computed using the following recursive equation. Basically, $T[0, 0, 0] = 1$, and $T[i, j, k] = 0$ if $i + j > k$. For other cases, $T[i, j, k]$ can be computed by Equation 6, where $\Pr(G_k \Rightarrow \emptyset)$ is the probability of G_k implicating a trivial certain graph $\emptyset$, and $\Pr(S \sqsubseteq_U G_k)$ is the probability of S occurring in G_k as given in Equation 5.

It is apparent that the φ-frequent probability of S can be computed from T by

$$\Pr(S) = \sum_{i=1}^{n} \sum_{j=0}^{\min\{\lfloor (1-\varphi)i/\varphi \rfloor,\, n-i\}} T[i, j, n]. \qquad (7)$$

Supposing that the exact values of $\Pr(G_i \Rightarrow \emptyset)$ and $\Pr(S \sqsubseteq_U G_i)$ for $1 \leq i \leq n$ are given as input, the procedure of the dynamic programming is presented in Figure 2. Obviously, the time complexity of the DP procedure is $\Theta(n^3)$.

Procedure DP
1. create array $T[0..n, 0..n, 0..n]$
2. $T[0, 0, 0] \leftarrow 1$
3. **for** $k \leftarrow 1$ to n **do**
4. **for** $j \leftarrow 0$ to k **do**
5. **for** $i \leftarrow 0$ to $k - j$ **do**
6. compute $T[i, j, k]$ using Equation 6
7. compute $\Pr(S)$ using Equation 7
8. **return** $\Pr(S)$

Figure 2: The dynamic programming procedure *DP*.

REMARK 1. *A similar dynamic programming method has previously been adopted by [11] to find frequent items over probabilistic data, in which the counterpart of* $\Pr(S \sqsubseteq_U G)$

$$T[i,j,k] = \begin{cases} \Pr(G_k \Rightarrow \emptyset) \cdot T[i,j,k-1] & \text{if } i=0 \text{ and } j=0, \\ \Pr(G_k \Rightarrow \emptyset) \cdot T[i,j,k-1] + \Pr(S \sqsubseteq_U G_k) \cdot T[i-1,j,k-1] & \text{if } i>0 \text{ and } j=0, \\ \Pr(G_k \Rightarrow \emptyset) \cdot T[i,j,k-1] & \\ \quad +(1-\Pr(G_k \Rightarrow \emptyset) - \Pr(S \sqsubseteq_U G_k)) \cdot T[i,j-1,k-1] & \text{if } i=0 \text{ and } j>0, \\ \Pr(G_k \Rightarrow \emptyset) \cdot T[i,j,k-1] + \Pr(S \sqsubseteq_U G_k) \cdot T[i-1,j,k-1] & \\ \quad +(1-\Pr(G_k \Rightarrow \emptyset) - \Pr(S \sqsubseteq_U G_k)) \cdot T[i,j-1,k-1] & \text{if } i>0 \text{ and } j>0. \end{cases} \quad (6)$$

Figure 1: Recursive equation for computing $T[i,j,k]$, where $0 \le i+j \le k$ and $1 \le k \le n$.

in the recursive equation is the probability that an item is contained in an x-tuple. Actually, the probability of an item being contained in an x-tuple can be evaluated in polynomial time. Unfortunately, the complexity result is negative for computing $\Pr(S \sqsubseteq_U G)$, which is shown in the following important theorem.

THEOREM 5. *It is #P-hard to compute the probability, $\Pr(S \sqsubseteq_U G)$, of a subgraph S occurring in an uncertain graph G.*

PROOF. The proof is very similar to the proof of Theorem 1 in [13], so it is omitted here. □

Although the time complexity of the *DP* procedure is $O(n^3)$, it is computationally prohibitive to evaluate the required input of the *DP* procedure.

5.2.2 Approximation by Intervals

The dynamic programming-based method is extended to an approximation algorithm that produce an interval $[p_l, p_u]$ in polynomial time such that $|p_u - p_l| \le \varepsilon$ and that $\Pr(S) \in [p_l, p_u]$ with probability at least $1-\delta$, where $0 < \delta < 1$.

To this end, an algorithm for estimating $\Pr(S \sqsubseteq_U G)$ with high accuracy is proposed and then is combined with the dynamic programming procedure *DP* to get the approximation algorithm for computing $[p_l, p_u]$. We clarify the details of each step as follows.

Estimation of $\Pr(S \sqsubseteq_U G)$. The fundamental idea of the algorithm for estimating $\Pr(S \sqsubseteq_U G)$ is to transform the problem of computing $\Pr(S \sqsubseteq_U G)$ to the problem of computing the probability of a DNF formula F being satisfied by a randomly and independently chosen truth assignment to the variables of F.

Given an uncertain graph G and a subgraph S, let us construct a DNF formula F first. Let $\mathbf{G}$ denote the certain graph obtained by removing uncertainties from G. As a side product of enumerating the children of the parent of S in the search tree, all subgraph isomorphisms from S to $\mathbf{G}$ have been obtained. Based on all these subgraph isomorphisms, all *embeddings*, $M_1, M_2, \ldots, M_m$, of S in $\mathbf{G}$ can be obtained. Each embedding M_i is a subgraph of $\mathbf{G}$ that S is subgraph isomorphic to. The DNF formula F therefore can be constructed as follows.

Step 1. Assign a distinct variable to each vertex contained in $M_1, M_2, \ldots, M_m$. The probability of the variable being assigned true is equal to the existence probability of the vertex in G.

Step 2. Assign a distinct variable to each edge contained in $M_1, M_2, \ldots, M_m$. The probability of the variable being assigned true is equal to the conditional existence probability of the edge in G.

Step 3. For each embedding M_i, construct a clause C_i by conjuncting all variables assigned to the vertices and edges in M_i.

Step 4. Build $F = C_1 \vee C_2 \vee \cdots \vee C_m$.

Let $\Pr(F)$ denote the probability of F being satisfied by a randomly and independently chosen truth assignment to the variables of F. It is easy to prove the following lemma.

LEMMA 2. $\Pr(S \sqsubseteq_U G) = \Pr(F)$.

The Monte-Carlo algorithm proposed in [5] can be applied to estimate $\Pr(F)$ within absolute error ε with probability at least $1-\delta$ for arbitrary $0 < \varepsilon, \delta < 1$.

Procedure ESTIMATE
1. **for all** vertex $v \in V(M_1) \cup V(M_2) \cup \cdots \cup V(M_m)$ **do**
2. assign variable x_v to v
3. **for all** edge $e \in E(M_1) \cup E(M_2) \cup \cdots \cup E(M_m)$ **do**
4. assign variable x_e to e
5. **for** $i \leftarrow 1$ to m **do**
6. $C_i \leftarrow \left(\bigwedge_{v \in V(M_i)} x_v\right) \wedge \left(\bigwedge_{e \in E(M_i)} x_e\right)$
7. $F \leftarrow C_1 \vee C_2 \vee \cdots \vee C_m$
8. estimate $\Pr(F)$ using the Monte-Carlo algorithm [5]
9. **return** the estimated value of $\Pr(F)$

Figure 3: The algorithm for estimating $\Pr(S \sqsubseteq_U G)$.

Figure 3 shows the procedure ESTIMATE of the algorithm for estimating $\Pr(S \sqsubseteq_U G)$. Let s be the number of edges of S. Since $|V(M_i)| \le |E(M_i)| = s$ for all $1 \le i \le n$, the DNF formula F can be constructed in $O(ms)$ time. Since F consists of m clauses and each clause contains $O(s)$ variables, the Monte-Carlo algorithm completes in $O\left(\frac{m^2 s}{\varepsilon^2} \log \frac{2}{\delta}\right)$ time [5]. Thus, the time complexity of the ESTIMATE procedure is $O\left(\frac{m^2 s}{\varepsilon^2} \log \frac{2}{\delta}\right)$. We have the following theorem.

THEOREM 6. *For any $0 < \varepsilon, \delta < 1$, $\Pr(S \sqsubseteq_U G)$ can be estimated by $\widehat{\Pr}(S \sqsubseteq_U G)$ such that*

$$\Pr\left(\left|\Pr(S \sqsubseteq_U G) - \widehat{\Pr}(S \sqsubseteq_U G)\right| \ge \varepsilon\right) \le \delta$$

in $O\left(\frac{m^2 s}{\varepsilon^2} \log \frac{2}{\delta}\right)$ time, where s is the number of edges of S, and m is the number of embeddings of S in the certain graph $\mathbf{G}$ obtained by removing uncertainties from G.

Approximation algorithm. Based on Theorem 6, an approximation algorithm for computing $[p_l, p_r]$ can be developed. Given an uncertain graph database $D = \{G_1, G_2, \ldots, G_n\}$, a subgraph S, an error tolerance $0 < \varepsilon < 1$ and a failure probability tolerance $0 < \delta < 1$, the algorithm works in a very simple and elegant way as follows.

Step 1. Compute $\Pr(G_i \Rightarrow \emptyset)$ by Equation 1 for $1 \le i \le n$.

Step 2. Estimate $\Pr(S \sqsubseteq_U G_i)$ by $\widehat{\Pr}(S \sqsubseteq_U G_i)$ within absolute error $\frac{\varepsilon}{2n}$ with probability at least $(1-\delta)^{1/n}$ for $1 \le i \le n$.

Step 3. Call the dynamic programming procedure DP with $\Pr(G_i \Rightarrow \emptyset)$ and $\widehat{\Pr}(S \sqsubseteq_U G_i)$ for $1 \le i \le n$ as input. Let $\widehat{\Pr}(S)$ be the output of DP.

Step 4. Return $[p_l, p_u] = [\widehat{\Pr}(S) - \varepsilon/2, \widehat{\Pr}(S) + \varepsilon/2]$.

Since the value of $\Pr(G_i \Rightarrow \emptyset)$ does not depend on the specific input subgraph S, the values of $\Pr(G_i \Rightarrow \emptyset)$ for all $1 \le i \le n$ can be computed just at the beginning of the mining algorithm and are reused in all subsequent calls of this approximation algorithm for all input subgraphs. The procedure APPROXIMATE of this approximation algorithm is presented in Figure 4.

Procedure APPROXIMATE
1. **for** $i \leftarrow 1$ to n **do**
2. $\quad \widehat{\Pr}(S \sqsubseteq_U G_i) \leftarrow$ the estimated value of $\Pr(S \sqsubseteq_U G_i)$
3. $\widehat{\Pr}(S) \leftarrow$ the output of the DP procedure using $\Pr(G_i \Rightarrow \emptyset)$ and $\widehat{\Pr}(S \sqsubseteq_U G_i)$ for $1 \le i \le n$ as input.
4. **return** $[\widehat{\Pr}(S) - \varepsilon/2, \widehat{\Pr}(S) + \varepsilon/2]$

Figure 4: The algorithm for approximating $\Pr(S)$.

The time complexity of the APPROXIMATE procedure is

$$O\left(\frac{4m^2n^3s}{\varepsilon^2}\log\frac{2}{1-(1-\delta)^{1/n}}\right),$$

where s is the number of edges of S, and m is the maximum number of embeddings of S in the uncertain graphs in D.

The expression of the time complexity can be simplified as follows. Let $f(x) = 1-(1-x)^{1/n}$ be a real function, where $0 \le x \le 1$ and $n \in \mathbb{N}$. By Taylor's expansion at $x = 0$,

$$\begin{aligned} f(x) &= f(0) + f'(0)\cdot x + \frac{f''(\xi)}{2}\cdot x^2 \\ &= 0 + \frac{x}{n} + \frac{1}{2n}(1-\frac{1}{n})(1-\xi)^{\frac{1}{n}-2}x^2, \end{aligned}$$

where $0 \le \xi \le x$. The Lagrange remainder is nonnegative, so $f(x) \ge x/n$. At $x = \delta$, we have $1-(1-\delta)^{1/n} \ge \delta/n$. Therefore, the time complexity of the APPROXIMATE procedure can be simplified to

$$O\left(\frac{4m^2n^3s}{\varepsilon^2}\log\frac{2n}{\delta}\right).$$

5.2.3 Theoretical Guarantees

It is obvious that the output $[p_l, p_u]$ of the APPROXIMATE procedure satisfies $|p_r - p_l| \le \varepsilon$. The rest of this subsection proves that the APPROXIMATE procedure fails with probability at most δ.

For ease of proof, let $DP(D, i)$ denote the output of the dynamic programming procedure DP over uncertain graph database $D = \{G_1, G_2, \ldots, G_n\}$ with $\widehat{\Pr}(S \sqsubseteq_U G_j)$ for $1 \le j \le i$ and $\Pr(S \sqsubseteq_U G_j)$ for $i+1 \le j \le n$ as input, where $\widehat{\Pr}(S \sqsubseteq_U G_j)$ is the estimated value of $\Pr(S \sqsubseteq_U G_j)$ within absolute error $\frac{\varepsilon}{2n}$ with probability at least $(1-\delta)^{1/n}$. We have the following crucial lemma.

LEMMA 3. *For $1 \le i \le n$,*

$$|DP(D, i-1) - DP(D, i)| \le |\Pr(S \sqsubseteq_U G_i) - \widehat{\Pr}(S \sqsubseteq_U G_i)|.$$

PROOF. Let DP' be the dynamic programming procedure that is the same as DP except that in line 7 the equation for computing the final result is replaced by

$$\sum_{i=1}^{n}\sum_{j=0}^{\min(\lfloor(1-\varphi)(i+1)/\varphi\rfloor,\, n-i)} T[i,j,n].$$

Also, let DP'' be the dynamic programming procedure that is the same as DP except that in line 7 the equation for computing the final result is substituted with

$$\sum_{i=1}^{n}\sum_{j=0}^{\min(\lfloor(1-\varphi)i/\varphi\rfloor-1,\, n-i)} T[i,j,n].$$

Furthermore, let $D' = D \setminus \{G_i\}$, $q_i = \Pr(G_i \Rightarrow \emptyset)$, $p_i = \Pr(S \sqsubseteq_U G_i)$, and $\widehat{p}_i = \widehat{\Pr}(S \sqsubseteq_U G_i)$. We have

$$\begin{aligned} DP(D, i-1) = q_i \cdot DP(D', i-1) + p_i \cdot DP'(D', i-1) \\ + (1 - q_i - p_i) \cdot DP''(D', i-1), \end{aligned}$$

and

$$\begin{aligned} DP(D, i) = q_i \cdot DP(D', i-1) + \widehat{p}_i \cdot DP'(D', i-1) \\ + (1 - q_i - \widehat{p}_i) \cdot DP''(D', i-1). \end{aligned}$$

By simple mathematics,

$$\begin{aligned} &|DP(D, i-1) - DP(D, i)| \\ &\quad = |p_i - \widehat{p}_i| \cdot |DP'(D', i-1) - DP''(D', i-1)|. \end{aligned}$$

Since the outputs of DP' and DP'' are both in $[0, 1]$,

$$|DP(D, i-1) - DP(D, i)| \le |p_i - \widehat{p}_i| \cdot 1 = |p_i - \widehat{p}_i|.$$

Thus, the lemma holds. □

Recall that $\widehat{\Pr}(S)$ is the estimated value of $\Pr(S)$ computed in line 3 of the APPROXIMATE procedure. We have the following theorem.

THEOREM 7. *With probability at least $1-\delta$,*

$$|\Pr(S) - \widehat{\Pr}(S)| \le \frac{\varepsilon}{2}.$$

PROOF. We observe that $\Pr(S) = DP(D, 0)$ and $\widehat{\Pr}(S) = DP(D, n)$. Then,

$$\begin{aligned} |\Pr(S) - \widehat{\Pr}(S)| &= |DP(D, 0) - DP(D, n)| \\ &= \left|\sum_{i=1}^{n}(DP(D, i-1) - DP(D, i))\right| \\ &\le \sum_{i=1}^{n}|DP(D, i-1) - DP(D, i)|. \end{aligned}$$

By Lemma 3,

$$|DP(D,i-1) - DP(D,i)| \leq |\Pr(S \sqsubseteq_U G_i) - \widehat{\Pr}(S \sqsubseteq_U G_i)| \leq \frac{\varepsilon}{2n}.$$

with probability at least $(1-\delta)^{1/n}$. Thus,

$$|\Pr(S) - \widehat{\Pr}(S)| \leq n \cdot \frac{\varepsilon}{2n} = \frac{\varepsilon}{2}.$$

with probability at least $\left((1-\delta)^{1/n}\right)^n = 1-\delta$. □

Consequently, the APPROXIMATE procedure fails with probability at most δ.

6. SETTING PARAMETERS

Let the input parameter δ of the APPROXIMATE procedure be set to the same value for all input subgraphs S. This section discusses how to set δ to guarantee the overall quality of approximation of the mining algorithm.

Note that, in most cases, it is expected to have $\tau > 0.5 > \varphi$. Then, we have the following crucial theorem.

THEOREM 8. *The probability of a frequent subgraph S being output as a mining result is at least $\left(\frac{1-\delta}{2}\right)^s$, where s is the number of edges of S.*

PROOF. Let $S_0, S_1, \ldots, S_s$ be the subgraphs on the path from the root to S in the search tree, where $S_0 = \emptyset$, $S_s = S$, and S_{i-1} is the parent of S_i for $1 \leq i \leq s$. Due to the Apriori property of φ-frequent probability, i.e., Lemma 1, all of $S_0, S_1, \ldots, S_{s-1}$ are also frequent. Furthermore, since subgraphs are visited in depth-first order, S is output as a mining result if and only if all of $S_0, S_1, \ldots, S_s$ are output as mining results. Mathematically,

$$\Pr(S \text{ is output}) = \Pr(S_0 \text{ is output}) \cdot \prod_{i=1}^{s} \Pr(S_i \text{ is output} \,|\, S_0, S_1, \ldots, S_{i-1} \text{ are output}).$$

Particularly, $\Pr(S_0 \text{ is output}) = 1$, and for $1 \leq i \leq s$,

$$\begin{aligned} &\Pr(S_i \text{ is output} \,|\, S_0, S_1, \ldots, S_{i-1} \text{ are output}) \\ =& \Pr(\widehat{\Pr}(S_i) \geq \varphi - \varepsilon/2) \\ =& \Pr(\Pr(S_i) - \widehat{\Pr}(S_i) \leq \Pr(S_i) - \varphi + \varepsilon/2). \end{aligned}$$

Due to the fact that $\Pr(S_i) \geq \tau > \varphi$ and to Theorem 7,

$$\begin{aligned} &\Pr(S_i \text{ is output} \,|\, S_0, S_1, \ldots, S_{i-1} \text{ are output}) \\ \geq& \Pr(\Pr(S_i) - \widehat{\Pr}(S_i) \leq \varepsilon/2) \\ \geq& (1-\delta)/2. \end{aligned}$$

Subsequently,

$$\Pr(S \text{ is output}) \geq \left(\frac{1-\delta}{2}\right)^s.$$

The theorem thus holds. □

Let $\ell_{\max}$ be the maximum number of edges of a frequent subgraph, which can either be estimated by sampling approaches [3] or be specified as a constraint on mining results [12]. From Theorem 8, we have the following corollary.

COROLLARY 1. *To ensure that the probability of any frequent subgraph being output is at least $1-\Delta$, the parameter δ should be at most $1 - 2 \cdot (1-\Delta)^{1/\ell_{\max}}$.*

Table 1: Summary of uncertain graph database.

organism	$\lvert V\rvert$	$\lvert E\rvert$	avg(P_E)
S. pombe (fission yeast)	162	300	0.148
D. melanogaster (fruit fly)	3751	7384	0.456
M. musculus (house mouse)	199	286	0.413
R. norvegicus (rat)	130	178	0.374
A. thaliana (thale cress)	513	1168	0.444
C. elegans (worm)	514	960	0.190

7. EXPERIMENTS

Extensive experiments were performed to evaluate the efficiency of the proposed algorithm and the quality of mining results. The experimental results are shown in this section.

7.1 Experimental Settings

We experimented on a real uncertain graph database that was obtained by integrating the BioGRID database[1] with the STRING database[2]. The real uncertain graph database contains the *protein-protein interaction* (PPI) *networks* of six organisms in the BioGRID database. A PPI network is an uncertain graph where vertices represent proteins, edges represent interactions between proteins, the labels of vertices are the COG functional annotations of proteins[3] provided by the STRING database, the existence probabilities of all vertices are 1, and the conditional existence probabilities of edges are provided by the STRING database. The summary of the uncertain graph database is shown in Table 1, where $|V|$ indicates the number of vertices, $|E|$ the number of edges, and avg(P_E) the average value of the conditional existence probabilities of edges.

The proposed algorithm was implemented in C on Linux. All experiments were performed on an IBM X3950 server with 2.4GHz Xeon E7440 CPU and 8GB of RAM. The operating system installed on the server is CentOS.

7.2 Experimental Results

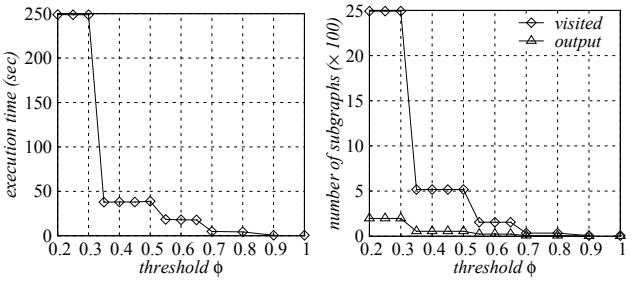
7.2.1 Experimental Results on Time Efficiency

We first performed experiments to test the execution time of the algorithm with respect to support threshold φ, confidence threshold τ and parameters ε and δ. The experimental results are presented and explained as follows.

Execution time vs. φ. Figure 5(a) shows the execution time of the algorithm while φ varies from 0.2 to 1, $\tau = 0.9$, $\varepsilon = 0.05$ and $\delta = 0.05$. It can be seen that the execution time decreases significantly as φ gets larger. This is because the number of frequent subgraphs reduces quickly as φ becomes larger, leading to the decrease in the number of visited subgraphs as shown in Figure 5(b).

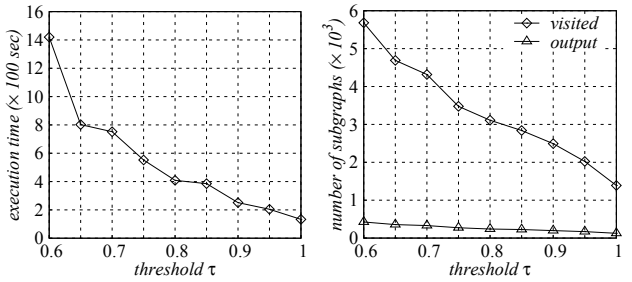
Execution time vs. τ. Figure 6(a) shows the execution time of the algorithm while τ varies from 0.6 to 1, $\varphi = 0.2$, $\varepsilon = 0.05$ and $\delta = 0.05$. The execution time decreases quickly as τ increases. The reason is that fewer subgraphs will have φ-frequent probability beyond τ as τ becomes larger, resulting in less subgraphs to be visited by the algorithm as shown in Figure 6(b).

[1] http://thebiogrid.org
[2] http://string-db.org
[3] http://www.ncbi.nih.gov/COG



(a) Execution time. (b) Number of subgraphs.

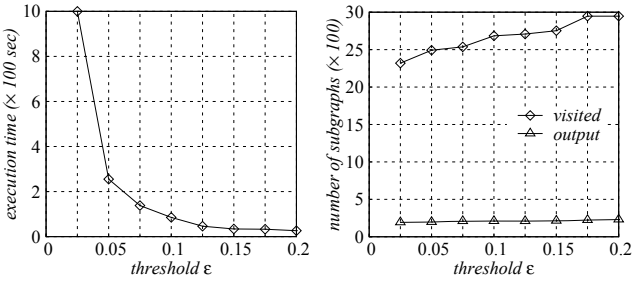
Figure 5: Impact of threshold φ on time efficiency.



(a) Execution time. (b) Number of subgraphs.

Figure 6: Impact of threshold τ on time efficiency.

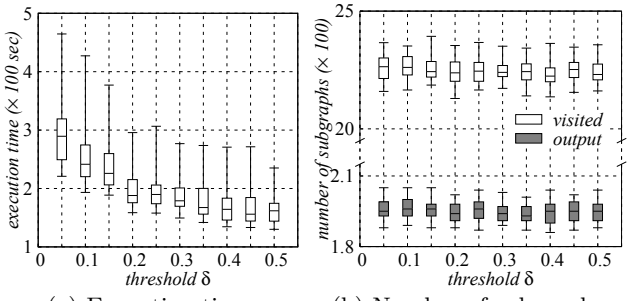
Execution time vs. ε. Figure 7(a) shows the execution time of the algorithm as ε varies from 0.025 to 0.2, $\varphi = 0.2$, $\tau = 0.9$ and $\delta = 0.05$. We can see that the execution time decreases substantially as ε increases. This is due to the following facts. The execution time of the algorithm is dominated by the time for computing the φ-frequent probabilities of all visited subgraphs. Although the number of visited subgraphs increases as ε gets larger as shown in Figure 7(b), the time for computing the φ-frequent probability of each subgraph is inversely proportional to ε^2 as analyzed in Section 5.2.2, thus decreasing the overall time for computing the φ-frequent probabilities of all visited subgraphs.



(a) Execution time. (b) Number of subgraphs.

Figure 7: Impact of parameter ε on time efficiency.

Execution time vs. δ. Unlike in the previous experiments with respect to φ, τ and ε, we can not figure out the relationship between the execution time and δ when running the algorithm only once for each tested value of δ. Hence, we ran the algorithm 30 times for each tested value of δ. Figure 8(a) illustrates the box plot of the execution time as δ varies from 0.05 to 0.5, $\varphi = 0.2$, $\tau = 0.9$ and $\varepsilon = 0.05$. The execution time statistically decreases as δ becomes larger. This is because the number of visited subgraphs does not vary significantly as shown in Figure 8(b) but the time for computing the φ-frequent probability of each subgraph is proportional to $\log(2n/\delta)$ as analyzed in Section 5.2.2.



(a) Execution time. (b) Number of subgraphs.

Figure 8: Impact of parameter δ on time efficiency.

7.2.2 Experimental Results on Quality of Approximation

We also evaluated the quality of approximation of the algorithm by testing the *precision* and *recall* of mining results. Precision is the proportion of frequent subgraphs in all output subgraphs, and recall is the proportion of output subgraphs in all frequent subgraphs.

Since it is computationally prohibitive to find all strictly frequent subgraphs, we instead use the set of subgraphs obtained for $\varphi = 0.2$, $\tau = 0.9$, $\varepsilon = 0.02$ and $\delta = 0.001$ as all frequent subgraphs with respect to $\varphi = 0.2$ and $\tau = 0.9$.

The experimental results with respect to parameters ε and δ are presented and analyzed as follows.

Precision vs. ε. Figure 9 shows the precision of the mining results while ε varies from 0.025 to 0.2, $\varphi = 0.2$, $\tau = 0.9$ and $\delta = 0.05$. We observe that the precision is over 90% for $\varepsilon \leq 0.05$ and goes down as ε gets larger. This is because with the increasing of ε, the number of infrequent subgraphs with φ-frequent probability within $[\tau - \varepsilon, \tau)$ increases, resulting in more infrequent subgraphs been output as results as illustrated in the #OIS row in Table 2. Moreover, since δ is fixed, the probability of a frequent subgraph being output is also fixed and is at least $\left(\frac{1-\delta}{2}\right)^s$ as proved in Theorem 8. Thus, the number of output frequent subgraphs is stable and is independent of ε as shown in the #OFS row in Table 2. The precision is equal to #OFS/(#OFS + #OIS), thus it decreases as ε becomes larger.

Recall vs. ε. By the arguments above, we know that the number of output frequent subgraphs #OFS is stable and is independent of ε. Moreover, for $\varphi = 0.2$ and $\tau = 0.9$, the number of frequent subgraphs is 184. Since the recall of the mining results is equal to #OFS/184, we have that the recall does not depend on ε and is over 94%.

Precision vs. δ. In this experiment, we ran the algorithm 30 times for each tested value of δ. Figure 10(a) shows the box plot of the precision of the mining results while δ varies from 0.05 to 0.5, $\varphi = 0.2$, $\varepsilon = 0.05$. It can be seen that

ε	0.025	0.05	0.075	0.1	0.125	0.15	0.175	0.2
#OFS	184	181	183	176	176	175	174	180
#OIS	8	16	25	33	34	39	49	50

Table 2: Number of output frequent subgraphs (#OFS) and number output infrequent subgraphs (#OIS) with respect to threshold ε.

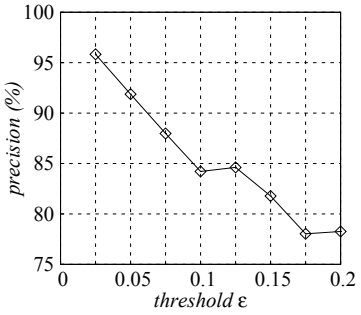
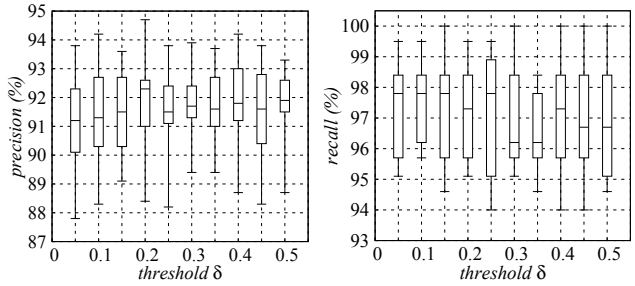


Figure 9: Impact of parameter ε on precision.

the precision is as high as 90% and does not depend on δ statistically. This is due to the following reasons. Since ε is fixed, the number of infrequent subgraphs with φ-frequent probability in $[\tau-\varepsilon,\tau)$ is also fixed. Although δ varies, the ratio of the probability of a specific frequent subgraph being output to the probability of a specific infrequent subgraph with φ-frequent probability in $[\tau-\varepsilon,\tau)$ being output is a constant. Therefore, the precision of the mining results is stable and is statistically independent of δ.



(a) Precision. (b) Recall.

Figure 10: Impact of δ on precision and recall.

Recall vs. δ. By Theorem 8, the probability of a frequent subgraph being output as a result is at least $\left(\frac{1-\delta}{2}\right)^s$, thus the recall of the mining results should decrease as δ increases. However, Figure 10(b) does not show this trend. The reason is that the Monte-Carlo algorithm [5] used in the ESTIMATE procedure for estimating the probability of a DNF formula being satisfied has much higher accuracy in practice than its theoretical lower bound used in the proof of Theorem 8.

8. CONCLUSIONS

In this paper, an approximate mining algorithm has been developed for efficiently and accurately mining frequent subgraphs over an uncertain graph database under probabilistic semantics. The algorithm guarantees to find any frequent subgraph with a provably high probability by carefully setting parameter δ using a systematic method. The extensive experiments on the real uncertain graph database verify that the algorithm is practically efficient and that the mining results have very high precision and recall.

Acknowledgements

The research work in this paper was in part supported by the NSF of China under Grant No. 60773063, the NSFC-RGC of China under Grant No. 60831160525, the National Grand Fundamental Research 973 Program of China under Grant No. 2006CB303000, and the NSF of China under Grant No. 60903017.

9. REFERENCES

[1] C. C. Aggarwal and H. Wang. *Managing and Mining Graph Data*. Springer, 2010.

[2] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Züfle. Probabilistic frequent itemset mining in uncertain databases. In *KDD*, pages 119–128, 2009.

[3] M. A. Hasan and M. J. Zaki. Output space sampling for graph patterns. *PVLDB*, 2(1):730–741, 2009.

[4] P. Hintsanen and H. Toivonen. Finding reliable subgraphs from large probabilistic graphs. *DMKD*, 17(1):3–23, 2008.

[5] R. M. Karp and M. Luby. Monte-Carlo algorithms for enumeration and reliability problems. In *FOCS*, pages 56–64, 1983.

[6] A. Kimmig and L. D. Raedt. Local query mining in a probabilistic prolog. In *IJCAI*, pages 1095–1100, 2009.

[7] M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1038–1051, 2004.

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

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

[10] G. Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *KDD*, pages 344–353, 2004.

[11] Q. Zhang, F. Li, and K. Yi. Finding frequent items in probabilistic data. In *SIGMOD*, pages 819–832, 2008.

[12] F. Zhu, X. Yan, J. Han, and P. S. Yu. gPrune: A constraint pushing framework for graph pattern mining. In *PAKDD*, pages 388–400, 2007.

[13] Z. Zou, J. Li, H. Gao, and S. Zhang. Frequent subgraph pattern mining on uncertain graph data. In *CIKM*, pages 583–592, 2009.

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

[15] Z. Zou, J. Li, H. Gao, and S. Zhang. Mining frequent subgraph patterns from uncertain graph data. *TKDE*, preprint, 2010.