Frequent Subgraph Pattern Mining on Uncertain Graph Data

Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang
Department of Computer Science and Technology
Harbin Institute of Technology
90 West Dazhi Street, Harbin, China
{znzou, lijzh, honggao, zhangshuocn}@hit.edu.cn

ABSTRACT

Graph data are subject to uncertainties in many applications due to incompleteness and imprecision of data. Mining uncertain graph data is semantically different from and computationally more challenging than mining exact graph data. This paper investigates the problem of mining frequent subgraph patterns from uncertain graph data. The frequent subgraph pattern mining problem is formalized by designing a new measure called expected support. An approximate mining algorithm is proposed to find an approximate set of frequent subgraph patterns by allowing an error tolerance on the expected supports of the discovered subgraph patterns. The algorithm uses an efficient approximation algorithm to determine whether a subgraph pattern can be output or not. The analytical and experimental results show that the algorithm is very efficient, accurate and scalable for large uncertain graph databases.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance

1 INTRODUCTION

Recently, *graph mining* has become an increasingly important research issue. Existing studies on graph mining are only concerned with *exact graphs* that are precise and complete. However, graph data are generally subject to uncertainties caused by noise, incompleteness and inaccuracy in practice. We call such kinds of graphs *uncertain graphs*.

Example 1. *In bioinformatics, interactions between proteins are generally represented as a graph, called* protein-protein interaction (PPI) network*, where vertices represent proteins, and edges represent interactions between proteins. Currently, there are large collections of PPIs detected by a variety of methods. It has been noted that all methods produce a significant amount of noisy interactions that don't really exist and miss a fraction of real interactions. Due to the inaccurate nature of PPI detection methods, it is more appropriate to represent a PPI network as an uncertain graph, where the uncertainty of each edge represents the chance of the interaction existing in reality [2]. Many methods have been proposed to derive uncertainties of protein interactions [20].*

Mining uncertain graph data is important in many applications. For example, [2] predicts the membership of a protein in a partially known protein complex by mining a PPI network as an uncertain graph; [6] models a wireless networks as an uncertain graph and extracts the most probable delivery subgraph to aid the design of routing protocols. In mining uncertain graph data, each discovered knowledge is associated with a confidence value computed from uncertainties to indicate the possibility of the knowledge existing in reality. Only knowledge occurring with high confidence can be regarded as useful.

As a central problem in graph mining, *frequent subgraph pattern mining* on exact graph data has gained a lot of attention [8, 14, 18, 27]. For uncertain graph data, frequent subgraph pattern mining is also a useful tool for analyzing uncertain graph data.

Example 2. *Biologists are often interested in identifying functional modules and evolutionarily conserved subnetworks from biological networks such as PPI networks. Frequent subgraph pattern mining has been shown to be an effective approach [13]. However, as shown above, biological networks are generally subject to uncertainties. So, it is important to find subgraph patterns that not only occur frequently in uncertain graphs but also have high confidence in terms of uncertainty to exist in reality.*

An *uncertain graph database* D containing 2 uncertain graphs, G_1 and G_2, is shown in Figure 1. The text on each vertex is the label of the vertex, the text on each edge is the label of the edge, and the real number on each edge is the *existence possibility* of the edge. The existence possibility of an edge means the possibility of the edge existing between the endpoints in an exact graph instance. Following the uncertain graph data model presented in Section 3, G_1 can be viewed as a succinct representation of 16 exact graphs, called *implicated graphs* in this paper, as shown in Figure 2. The probability distribution over the 16 implicated graphs is derived from the existence possibilities of the edges in G_1. Similarly, G_2 represent 8 implicated graphs. By combining any implicated graph of G_1 with any implicated graph of G_2, we obtain an exact graph database, called an *implicated graph database*. Due to the various combinations of implicated graphs, D has totally $16 \times 8 = 128$ implicated graph databases. Following this model, a subgraph pattern is an exact graph that is contained in at least one implicated graph databases. Under this new model, the significance of a subgraph pattern should be determined by considering both the number of occurrences of the subgraph patterns in the implicated graph databases and the probabilities of the implicated graph databases.

This paper investigates the problem of mining frequent subgraph patterns on uncertain graph data. The problem poses several new

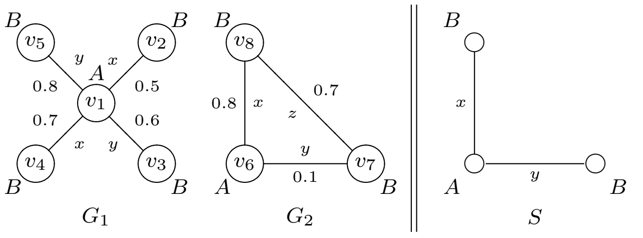


Figure 1: An example of uncertain graph database $D = \{G_1, G_2\}$ and subgraph pattern S.

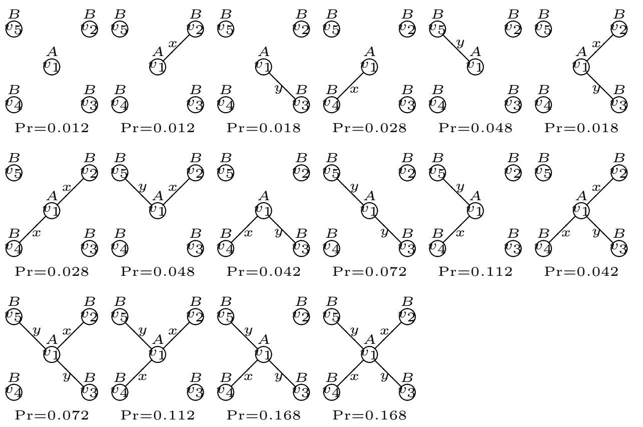


Figure 2: The probability distribution of all implicated graphs of uncertain graph G_1 in Figure 1.

challenges. On exact graph data, the significance of a subgraph pattern is measured by *support*, i.e. the proportion of graphs containing the subgraph pattern. However, such definition doesn't make sense on uncertain graph data because the containment relationship between uncertain graphs is uncertain. To address this challenge in semantics, the support of a subgraph pattern S in an uncertain graph database D should be defined as a *random variable* over the support values of S in all implicated graph databases of D. Then, the significance of S can be measured by the expected value, called *expected support*, of the support values of S in all implicated graph databases of D. If the expected support of S is no less than a threshold specified by users, then S is *frequent*. Therefore, the frequent subgraph pattern mining problem can be stated as follows. *Given an uncertain graph database D and an expected support threshold, find all frequent subgraph patterns in D.*

Discovering all frequent subgraph patterns in an uncertain graph database is a very challenging problem. Firstly, we prove that it is *#P-complete* [22] to compute the expected support of a subgraph pattern in an uncertain graph database, i.e. there exist no efficient algorithms to determine whether a subgraph pattern is frequent or not. Secondly, the number of all subgraph patterns in an uncertain graph database is extremely large in general, and it is even #P-hard to count the number of all subgraph patterns. So, it is unaffordable to examine all of them to find the frequent ones.

Due to the hardness of the problem, an approximate mining algorithm, called *MUSE* (Mining Uncertain Subgraph pattErns), is proposed to find an approximate set of frequent subgraph patterns in an uncertain graph database. It approximates the set of all frequent subgraph patterns in the following manner. Let $minsup$ be the expected support threshold and $\varepsilon \in [0, 1]$ be a *relative error tolerance*. In *MUSE*, all subgraph patterns with expected support at least $minsup$ are output, but all subgraph patterns with expected support less than $(1 - \varepsilon)minsup$ are not output. Moreover, decisions are arbitrary for subgraph patterns with expected support in $[(1 - \varepsilon)minsup, minsup)$.

The *MUSE* algorithm adopts two critical techniques. The first one is the efficient method to determine whether a subgraph pattern can be output or not. It first approximates the expected support of a subgraph pattern by an interval enclosing the expected support of the subgraph pattern and then makes decision on whether the subgraph pattern can be output or not by checking the overlapping relationship between the approximated interval and $[(1 - \varepsilon) \cdot minsup, minsup)$. In this way, it avoids the difficulty in exactly computing the expected support of the subgraph pattern. The second technique is the efficient method to examine subgraph patterns. We prove that the expected support satisfies the *apriori property*, that is, all supergraphs of an infrequent subgraph patterns are also infrequent. To take advantage of this property, all subgraph patterns are organized into a tree, and the tree is traversed in the depth-first strategy. If a subgraph pattern can not be output as a result, then all its descendants in the tree need not to be examined.

Extensive experiments were carried out to evaluate the efficiency, approximation quality and scalability of *MUSE* and the impact of uncertainties on the efficiency of *MUSE*. The analysis and the experimental results show that *MUSE* is very efficient, accurate and scalable on large uncertain graph databases.

2 RELATED WORK

A number of algorithms have been proposed to discover frequent subgraph patterns from exact graph data [8, 10, 14, 18, 23, 24, 27]. To reduce the number of redundant subgraph patterns, [28] proposed CloseGraph to discover frequent closed subgraph patterns, [9] developed SPIN to discover frequent maximal subgraph patterns, and [15] presented the RP-GD and RP-FP algorithms to summarize subgraph patterns. In addition, some variants of the frequent subgraph pattern mining problem have been studied, such as the discovery of frequent closed cliques [25], frequent closed quasi-cliques [29], cross-graph quasi-cliques [19], correlated subgraph patterns [12] and significant subgraph patterns [26]. However, all these algorithms are designed only for mining exact graph data and can not be extended to uncertain graph data.

Existing work on mining uncertain data has focused on clustering [4], frequent item/itemset mining [1, 3, 30], classification [21], and so on. However, all the algorithms investigate mining structured data in uncertain relational data models rather than mining uncertain graph data and can not be shifted to uncertain graph data mining.

To the best of our knowledge, there is no literature to date on mining frequent subgraph patterns from uncertain graph data. This paper is the first one to investigate this problem.

3 PROBLEM STATEMENT

In this paper, the vertex set and edge set of a graph G are denoted by $V(G)$ and $E(G)$, respectively.

3.1 Model of Uncertain Graphs

Definition 1. An *uncertain graph* is a system $G = ((V, E), \Sigma, L, P)$, where (V, E) is an undirected graph, Σ is a set of labels, $L : V \cup E \to \Sigma$ is a function assigning labels to vertices and edges, and $P : E \to (0, 1]$ is a function assigning existence possibility values to edges.

The *existence possibility*, $P((u, v))$, of an edge (u, v) is the possibility of the edge existing between vertices u and v. Specifically, $P((u, v)) = 1$ indicates that edge (u, v) definitely exists. Thus, an *exact graph*[1] is a special uncertain graph with existence possibilities of 1 on all edges. Unlike an exact graph, an uncertain graph *implicates* a set of exact graphs. Formally, an exact graph $I = ((V', E'), \Sigma', L')$ is an *implicated graph* of an uncertain graph $G = ((V, E), \Sigma, L, P)$, denoted by $G \Rightarrow I$, if and only if $V' = V$, $E' \subseteq E$, $\Sigma' \subseteq \Sigma$ and $L' = L|_{V' \cup E'}$, where $L|_{V' \cup E'}$ is the function obtained by restricting L to $V' \cup E'$. For simplicity, we assume that all existence possibilities of edges are *independent*. The independence assumption has been shown to be reasonable in many real applications [2, 6, 7]. Based on this assumption, the possibility of an uncertain graph G implicating an exact graph I is

$$P(G \Rightarrow I) = \prod_{e \in E(I)} P(e) \prod_{e' \in E(G) \setminus E(I)} (1 - P(e')), \quad (1)$$

where $P(e)$ is the existence possibility of edge e. Eq. (1) holds because all edges in $E(I)$ are included in I but all in $E(G) \setminus E(I)$ are not included in I. Let $I(G)$ denote the set of all implicated graphs of an uncertain graph G. Apparently, $|I(G)| = 2^{|E(G)|}$. Moreover, it is easy to show that function $P(G \Rightarrow I)$ defines a probability distribution over $I(G)$.

An *uncertain graph database* is a set of uncertain graphs. It essentially represents a set of *implicated graph databases*. Formally, an implicated graph database of an uncertain graph database $D = \{G_1, G_2, \ldots, G_n\}$ is a set of exact graphs $d = \{I_1, I_2, \ldots, I_n\}$ such that $G_i \Rightarrow I_i$ for $1 \leq i \leq n$. The set of all implicated graph databases of D is denoted by $I(D)$. Obviously, $|I(D)| = \prod_{i=1}^{n} 2^{|E(G_i)|}$. Assuming that the uncertain graphs in an uncertain graph database are mutually independent, the possibility of a graph database $d = \{I_1, I_2, \ldots, I_n\}$ being implicated by an uncertain graph database $D = \{G_1, G_2, \ldots, G_n\}$ is

$$P(D \Rightarrow d) = \prod_{i=1}^{n} P(G_i \Rightarrow I_i), \quad (2)$$

where $P(G_i \Rightarrow I_i)$ is the probability of G_i implicating I_i. It is easy to prove that function $P(D \Rightarrow d)$ defines a probability distribution over $I(D)$.

Example 3. *Figure 1 shows an example of uncertain graph database $D = \{G_1, G_2\}$. G_1 represents the probability distribution over the 16 implicated graphs of G_1 as shown in Figure 2. G_2 represents the probability distribution over the 8 implicated graphs of G_2. D represents the probability distribution over the 128 implicated graph databases of D.*

3.2 Problem Definition

Definition 2. An exact graph $G = (V, E, \Sigma, L)$ is *subgraph isomorphic* to another exact graph $G' = (V', E', \Sigma', L')$, denoted by $G \sqsubseteq_{ex} G'$, if there exists an injection $f : V \to V'$ such that (1) $L(v) = L'(f(v))$ for any $v \in V$, (2) $(f(u), f(v)) \in E'$ for any $(u, v) \in E$, and (3) $L((u, v)) = L'((f(u), f(v)))$ for any $(u, v) \in E$. The injection f is called a *subgraph isomorphism* from G to G'. The subgraph (V'', E'') of G' with $V'' = \{f(v)|v \in V\}$ and $E'' = \{(f(u), f(v))|(u, v) \in E\}$ is called the *embedding* of G in G' under f.

In conventional frequent subgraph pattern mining, a *subgraph pattern* is defined as a connected subgraph that is subgraph isomorphic to at least one exact graph in the input exact graph database, and the *support* of a subgraph pattern S in an exact graph database D is defined as $sup_D(S) = \frac{|\{G|\ S \sqsubseteq_{ex} G \text{ and } G \in D\}|}{|D|}$. However, such concepts don't make sense in uncertain graph databases since an exact subgraph is embedded in an uncertain graph in a probabilistic sense. Hence, these concepts should be redefined in the context of uncertain graph databases.

In an uncertain graph database D, a *subgraph pattern* is a connected exact graph that is subgraph isomorphic to at least one implicated graph in some implicated graph database of D. Let S and S' be two subgraph patterns in D. We call S a *subpattern* of S', or S' a *superpattern* of S, if $S \sqsubseteq_{ex} S'$, and call S a *direct subpattern* of S', or S' a *direct superpattern* of S, if $S \sqsubseteq_{ex} S'$ and $|E(S)| + 1 = |E(S')|$.

Let $I(D)$ be the set of all implicated graph databases of D. The *support* of a subgraph pattern S in D is a random variable over $I(D)$ with probability distribution

s_1	s_2	$\cdots$	s_m
$P(s_1)$	$P(s_2)$	$\cdots$	$P(s_m)$

where $m = |\{sup_d(S)|d \in I(D)\}|$, $s_i = sup_d(S)$ is the conventional support of S in an implicated graph database $d \in I(D)$, and $P(s_i) = \sum_{d \in I(D) \text{ and } sup_d(S) = s_i} P(D \Rightarrow d)$ is the probability of support value s_i for $1 \leq i \leq m$.

The significance of the subgraph pattern S can be measured by the expected value of the support of S as defined above, called the *expected support* of S in D, i.e.

$$esup_D(S) = \sum_{i=1}^{m} s_i P(s_i) = \sum_{d \in I(D)} sup_d(S) P(D \Rightarrow d). \quad (3)$$

A subgraph pattern S is *frequent* in an uncertain graph database D if the expected support of S in D is not less than a user-specified threshold $minsup \in [0, 1]$. Then, the problem of *discovering frequent subgraph patterns on an uncertain graph database* can be defined as follows.
Input: an uncertain graph database D and an expected support threshold $minsup$.
Output: the set of all frequent subgraph patterns in D, i.e. $\{S|\ S$ is a subgraph pattern in D and $esup_D(S) \geq minsup\}$.

4 COMPLEXITY OF THE FREQUENT SUBGRAPH PATTERN MINING PROBLEM

Before proving the computational complexity of the frequent subgraph pattern mining problem, we first reformulate the expected support measure.

Given an uncertain graph database D, a subgraph pattern S in D is said to *occur* in an uncertain graph $G \in D$, denoted by $S \sqsubseteq_U G$, if S is subgraph isomorphic to at least one implicated graph of G. The probability of S occurring in G is

$$P(S \sqsubseteq_U G) = \sum_{I \in I(G)} P(G \Rightarrow I) \psi(I, S), \quad (4)$$

where $I(G)$ is the set of all implicated graphs of G, and $\psi(I, S) = 1$ if S is subgraph isomorphic to I and $\psi(I, S) = 0$ otherwise. Then, Eq. (3) can be rewritten as follows.

$$\begin{aligned} esup_D(S) &= \sum_{d \in I(D)} sup_d(S) P(D \Rightarrow d) \\ &= \sum_{d = \{I_1, I_2, \ldots, I_{|D|}\} \in I(D)} \left(\frac{P(D \Rightarrow d)}{|D|} \sum_{i=1}^{|D|} \psi(I_i, S) \right) \\ &= \frac{1}{|D|} \sum_{i=1}^{|D|} \sum_{I \in I(G_i)} \psi(I, S) P(G_i \Rightarrow I) \\ &= \frac{1}{|D|} \sum_{i=1}^{|D|} P(S \sqsubseteq_U G_i). \end{aligned} \quad (5)$$

[1]A conventional labeled graph [14, 27] is called an exact graph in this paper, which is a 3-tuple $G = ((V, E), \Sigma, L)$, where (V, E) is an undirected graph, Σ is a set of labels, and $L : V \cup E \to \Sigma$ is a labeling function of the vertices and edges.

Hence, the expected support of S in D can be efficiently computed using Eq. (5) instead of using Eq. (3).

Theorem 1. *It is #P-complete to compute the probability of a subgraph pattern occurring in an uncertain graph.*

Proof. We prove the theorem by reducing the #P-complete *DNF counting problem* [22] to the problem of computing the probability, $P(S \sqsubseteq_U G)$, of a subgraph pattern S occurring in an uncertain graph G.

The DNF counting problem can be formulated as follows. Let $F = C_1 \vee C_2 \vee \cdots \vee C_n$ be a boolean formula in *disjunctive normal form (DNF)* on m boolean variables $x_1, x_2, \ldots, x_m$. Each clause C_i is of the form $C_i = l_1 \wedge l_2 \wedge \cdots \wedge l_k$, where l_j is a boolean variable in $\{x_1, x_2, \ldots, x_m\}$. Let $\Pr(x_i)$ be the probability of x_i being assigned true. The DNF counting problem is to compute the probability of F being satisfied by a randomly and independently chosen truth assignment to the variables, denoted by $\Pr(F)$. Given an instance of the DNF counting problem, an instance of the problem of computing $P(S \sqsubseteq_U G)$ can be constructed as follows.

First, construct an uncertain graph G. The vertex set of G is $V(G) = \{c_1, c_2, \ldots, c_n, u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_m\}$. The labels of $c_1, c_2, \ldots, c_n$ are α, and the labels of $u_1, u_2, \ldots, u_m$ and $v_1, v_2, \ldots, v_m$ are β. The edge set of G is constructed as follows. For each variable x_i in the DNF formula F, add an edge (u_i, v_i) associated with existence possibility of $\Pr(x_i)$ to G. For each variable x_j in each clause C_i, add an edge (c_i, u_j) associated with existence possibility of 1 to G. All edges of G are labeled γ.

Next, construct a subgraph pattern S. The vertex set of S is $V(S) = \{c', u'_1, u'_2, \ldots, u'_k, v'_1, v'_2, \ldots, v'_k\}$. The label of c' is α, and the labels of $u'_1, u'_2, \ldots, u'_k$ and $v'_1, v'_2, \ldots, v'_k$ are β. The edge set of S is $E(S) = \{(c', u'_1), (c', u'_2), \ldots, (c', u'_k), (u'_1, v'_1), (u'_2, v'_2), \ldots, (u'_k, v'_k)\}$. All edges of S are labeled γ.

For example, given a DNF formula $(x_1 \wedge x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge x_4)$ and the probabilities $\Pr(x_1), \Pr(x_2), \ldots, \Pr(x_4)$ of the variables being assigned true, the uncertain graph G and the subgraph pattern S constructed from the DNF formula are shown in Figure 3, where the labels of the edges are omitted for clarity.

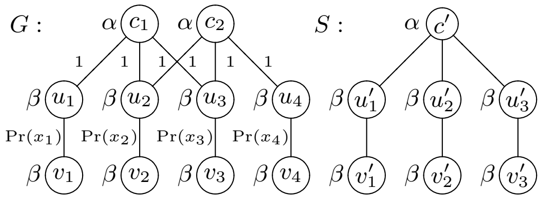


Figure 3: The uncertain graph G and subgraph pattern S constructed for $(x_1 \wedge x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge x_4)$.

Each truth assignment to the variables in F one-to-one corresponds to an implicated graph of G, i.e. edge (u_i, v_i) exists in the implicated graph if and only if x_i = true. The probability of each truth assignment is equal to the probability of the implicated graph that the truth assignment corresponds to. A truth assignment satisfies F if and only if the implicated graph that the truth assignment corresponds to contains subgraph pattern S. Thus, $\Pr(F)$ is equal to the probability, $P(S \sqsubseteq_U G)$, of S occurring in G. This completes the polynomial time reduction. □

By Theorem 1 and Eq. (5), we obtain the following corollary.

Corollary 1. *It is #P-complete to compute the expected support of a subgraph pattern in an uncertain graph database.*

The number of frequent subgraph patterns in an uncertain graph database is generally exponential to the size of the uncertain graph database. Naturally, the complexity of any algorithm to mine frequent subgraph patterns is exponential with respect to the size of the input. More formally, we have the following theorem.

Theorem 2. *The problem of counting the number of frequent subgraph patterns in an uncertain graph database for an arbitrary expected support threshold is #P-hard.*

Proof. We prove the theorem by giving a polynomial time reduction from the #P-complete problem of counting the number of satisfying truth assignments of *a monotone k-CNF formula* [22] to the problem of counting the number of frequent subgraph patterns in an uncertain graph database. A monotone k-CNF formula is a boolean formula in conjunctive normal form (CNF) in which every clause has at most k literals and every literal is not negated.

Let $F = D_1 \vee D_2 \vee \cdots \vee D_n$ be a monotone k-CNF formula on m boolean variables $x_1, x_2, \ldots, x_m$. Each clause D_i is of the form $D_i = l_1 \wedge l_2 \wedge \cdots \wedge l_{r_i}$, where each l_j is an unengaged boolean variable and $r_i \leq k$. An uncertain graph database D can be constructed as follows. For each clause $D_i = l_1 \wedge l_2 \wedge \cdots \wedge l_{r_i}$ in F, construct an uncertain graph G_i. The vertex set of G_i is $V(G_i) = \{v_0^i, v_1^i, \ldots, v_{m-r_i}^i\}$. The edge set of G_i is $E(G_i) = \{(v_0^i, v_1^i), (v_0^i, v_2^i), \ldots, (v_0^i, v_{m-r_i}^i)\}$. All vertices of G_i are labeled α. Each edge of G_i is associated with a distinct label in $\{x_1, x_2, \ldots, x_m\} \setminus \{l_1, l_2, \ldots, l_{r_i}\}$. Each edge of G_i has existence possibility of 1. For example, given a monotone 2-CNF formula $(x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge x_4$, the uncertain graph database D constructed is shown in Figure 4.

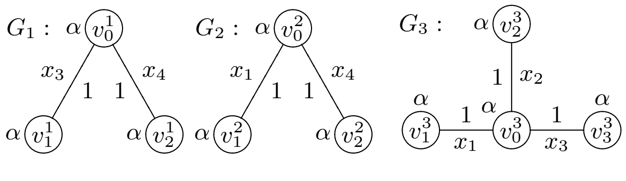


Figure 4: The uncertain graph database, $D = \{G_1, G_2, G_3\}$, constructed for $(x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge x_4$.

We establish the correspondence between the number of satisfying truth assignments of F and the number of frequent subgraph patterns in D. Each truth assignment π to the variables in F one-to-one corresponds to an exact graph g_π. In particular, suppose the variables in π assigned true are $x_1, x_2, \ldots, x_l$. The vertex set of g_π is $\{v_0, v_1, \ldots, v_l\}$. The edge set of g_π is $\{(v_0, v_1), (v_0, v_2), \ldots, (v_0, v_l)\}$. All vertices of g_π are labeled α. The edge (v_0, v_i) is labeled x_i for $1 \leq i \leq l$. Note that the expected support of a subgraph pattern in D is identical to the traditional support of the subgraph pattern in D since D is an exact graph database at this time. A truth assignment π doesn't satisfy F if and only if the exact graph, g_π, corresponding to π is a frequent subgraph pattern in D with respect to threshold $1/n$. Thus, the number of frequent subgraph patterns in D is 2^m minus the number of satisfying truth assignments of F. This completes the polynomial time reduction. □

From the #P-hardness of the problem of counting the number of frequent subgraph patterns, the NP-hardness of the problem of finding all frequent subgraph patterns can be readily derived [5].

5 APPROXIMATE MINING ALGORITHM

5.1 Overview of the Algorithm

Due to the NP-hardness of the frequent subgraph pattern mining problem, an approximate mining algorithm is proposed to find an approximate set of frequent subgraph patterns. More formally, let *minsup* be the input expected support threshold and $\varepsilon \in [0, 1]$ be a *relative error tolerance*. All subgraph patterns with expected support at least $minsup$ will be output, but all subgraph patterns with expected support less than $(1-\varepsilon)minsup$ will not be output. Decisions are arbitrary for subgraph patterns with expected support in $[(1-\varepsilon)minsup, minsup)$.

The approximate mining algorithm has two main objectives.

1. Determine as efficiently as possible whether a subgraph pattern can be output or not.

2. Examine the subgraph patterns as efficiently as possible to find the frequent ones.

5.1.1 Method to Complete Objective I

To complete the first objective, we approximate the expected support, $esup_D(S)$, of a subgraph pattern S in the uncertain graph database D by a closed interval, denoted $[\underline{esup}_D(S), \overline{esup}_D(S)]$, such that $esup_D(S) \in [\underline{esup}_D(S), \overline{esup}_D(S)]$ and then determine whether S can be output or not by testing the following conditions.

Condition 1. If $\overline{esup}_D(S) \geq minsup$ and $\underline{esup}_D(S) \geq (1-\varepsilon)minsup$, then output S since it is certain that $esup_D(S) \geq (1-\varepsilon)minsup$ and it is probable that $esup_D(S) > minsup$. This condition is illustrated on the top of Figure 5.

Condition 2. If $\overline{esup}_D(S) < minsup$, then don't output S since it is certain that $esup_D(S) < minsup$. This condition is illustrated in the middle of Figure 5.

Condition 3. If $\overline{esup}_D(S) \geq minsup$ and $\underline{esup}_D(S) < (1-\varepsilon)minsup$, then approximate $esup_D(S)$ by a smaller interval and test the conditions again since we are unable to decide whether $esup_D(S) > minsup$ or $esup_D(S) < (1-\varepsilon)minsup$ using the current interval. This condition is illustrated at the bottom of Figure 5.

It is interesting to note that if the width of the interval $[\underline{esup}_D(S), \overline{esup}_D(S)]$ is less than $\varepsilon \cdot minsup$, then either condition 1 or condition 2 will be satisfied. For this reason, it is sufficient to approximate $esup_D(S)$ by an interval with width at most $\varepsilon \cdot minsup$.

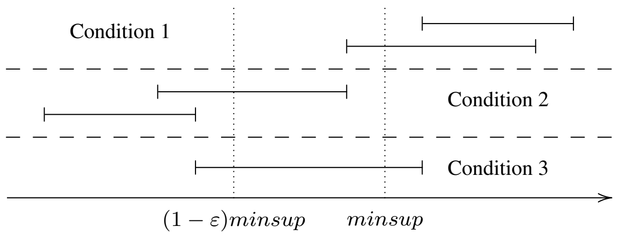


Figure 5: Illustrations of conditions for deciding whether to output a subgraph pattern or not.

5.1.2 Method to Complete Objective II

To fulfill the second objective, we first study the property of the expected support measure. For any uncertain graph $G \in D$ and any subgraph patterns S and S' in D, if S is a subpattern of S', then $\psi(I, S) \geq \psi(I, S')$ for any implicated graph I of G, where $\psi(I, S) = 1$ if S is subgraph isomorphic to I and $\psi(I, S) = 0$ otherwise. Then, we have $P(S \sqsubseteq_U G) \geq P(S' \sqsubseteq_U G)$ by Eq. (4). This is called the *apriori property* of the probability of a subgraph pattern occurring in G. Following this property, we also have $esup_D(S) \geq esup_D(S')$. This is called the apriori property of the expected support measure. A direct conclusion from the apriori property is that all subpatterns of a frequent subgraph pattern are also frequent, and all superpatterns of an infrequent subgraph pattern are also infrequent. This result can be utilized to reduce the complexity of the mining algorithm.

Then, we organize all subgraph patterns in the uncertain graph database by a structure and search the structure systematically to find all frequent subgraph patterns by taking advantage of the apriori property of the expected support. Based on the direct subpattern relationship, all subgraph patterns in the uncertain graph database D can be organized as a directed acyclic graph (DAG) with nodes representing subgraph patterns, and edges representing direct subpattern relationships. Figure 6 shows the DAG of the subgraph patterns in the uncertain graph database D in Figure 1. In a DAG of subgraph patterns, a subgraph pattern may have more than one parent. By requiring each subgraph pattern, except those having no parents, to keep only one parent using some specific schemes, the DAG can be simplified to a tree. A number of such schemes have been proposed [8, 10, 14, 18, 27]. For example, using the DFS coding scheme [27], the DAG in Figure 6 can be simplified to the tree highlighted by the solid directed arcs in Figure 6. We call such tree a *search tree* of subgraph patterns. The advantage of organizing subgraph patterns into a search tree is that if a subgraph pattern is known to be infrequent, then all its descendants in the search tree can be pruned due to the apriori property of the expected support.

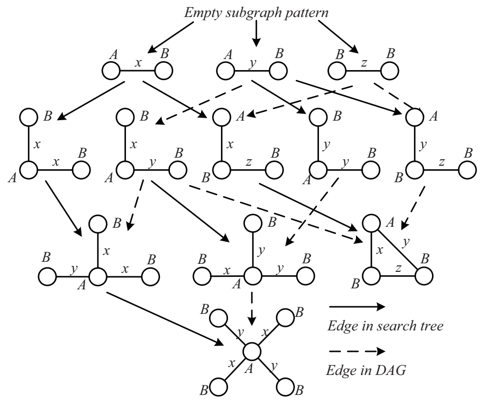


Figure 6: The search tree of the subgraph patterns in the uncertain graph database D in Figure 1.

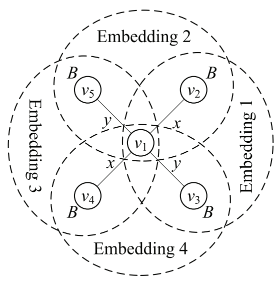
Thus, the problem of mining frequent subgraph patterns from an uncertain graph database is to traverse the search tree to find all frequent subgraph patterns with low computational complexity. The proposed approximate mining algorithm employs depth-first strategy to traverse the search tree. It works as follows.

Step 1. Let T be an empty stack. Scan the edges of the uncertain graphs in D to get all subgraph patterns consisting of only one edge, and push them into T.

Step 2. Pop the subgraph pattern S on the top of T. Find the subgraph isomorphisms from S to every uncertain graph $G \in D$ and get the embeddings of S in G under the subgraph isomorphisms just found. Approximate the expected support, $esup_D(S)$, of S in D by an interval, denoted $[\underline{esup}_D(S), \overline{esup}_D(S)]$, such that $esup_D(S) \in [\underline{esup}_D(S), \overline{esup}_D(S)]$ and that the width of $[\underline{esup}_D(S), \overline{esup}_D(S)]$ is at most $\varepsilon \cdot minsup$.

Step 3. Determine whether S can be output or not by testing conditions 1 and 2 given above using $[\underline{esup}_D(S), \overline{esup}_D(S)]$. If S can not be output, then the subtree rooted at S can be pruned due to the apriori property of the expected support, thus we skip this step and go to step 4. If S can be output, then output S and generate all direct superpatterns of S based on the embeddings of S in the uncertain graphs in D. For each generated superpattern S', if S' is a child of S in the search tree, then push S' into T, otherwise S' must be a child of another subgraph pattern S'' and should not be examined in the subtree rooted at S.

Step 4. If $T = \emptyset$, then terminate, otherwise go to step 2.



edge	var.	prob.
(v_1, v_2)	x_1	0.5
(v_1, v_3)	x_2	0.6
(v_1, v_4)	x_3	0.7
(v_1, v_5)	x_4	0.8

embed.	clause
1	$C_1 = x_1 \wedge x_2$
2	$C_2 = x_1 \wedge x_4$
3	$C_3 = x_3 \wedge x_4$
4	$C_4 = x_2 \wedge x_3$

$F = C_1 \vee C_2 \vee C_3 \vee C_4$

	C_1	C_2	C_3	C_4	total
Pr	0.3	0.4	0.56	0.42	1.68

	$C_1 \wedge C_2$	$C_1 \wedge C_3$	$C_1 \wedge C_4$	$C_2 \wedge C_3$	$C_2 \wedge C_4$	$C_3 \wedge C_4$	total
Pr	0.24	0.168	0.21	0.28	0.168	0.336	1.402

	$C_1 \wedge C_2 \wedge C_3$	$C_1 \wedge C_2 \wedge C_4$	$C_1 \wedge C_3 \wedge C_4$	$C_2 \wedge C_3 \wedge C_4$	total
Pr	0.168	0.168	0.168	0.168	0.672

	$C_1 \wedge C_2 \wedge C_3 \wedge C_4$	total
Pr	0.168	0.168

$\Pr(F) = 1.68 - 1.402 + 0.672 - 0.168 = 0.782.$

Figure 7: A running example of the exact algorithm for computing $P(S \sqsubseteq_U G)$.

As can be seen from the discussion above, the approximation of expected supports is substantial for reducing the complexity of the mining algorithm. In the rest of this section, we propose efficient algorithms to compute expected supports.

5.2 Algorithm for Computing Expected Supports

Eq. (5) shows that the expected support of a subgraph pattern S in an uncertain graph database D can be computed by averaging the probability of S occurring in every uncertain graph $G \in D$, i.e. $P(S \sqsubseteq_U G)$. However, it is #P-complete to compute $P(S \sqsubseteq_U G)$. To overcome the difficulty, we propose an optimized exact algorithm to exactly compute $P(S \sqsubseteq_U G)$ for small instances of the problem and an approximation algorithm to approximate $P(S \sqsubseteq_U G)$ for large instances of the problem.

5.2.1 Fundamental Technique

To compute $P(S \sqsubseteq_U G)$ exactly based on its definition, i.e. Eq. (4), we must compute the probability distribution over all $2^{|E(G)|}$ implicated graphs of G and perform $2^{|E(G)|}$ subgraph isomorphism testings from S to all implicated graphs of G, which is intractable even if G is of moderate size. Note that this naive method can't scale if G has more than 30 edges in practice. In this paper, we develop a new approach to compute $P(S \sqsubseteq_U G)$ based on the embeddings of S in G.[2]

The fundamental technique of the new approach is to transform the problem of computing $P(S \sqsubseteq_U G)$ to the DNF counting problem. Let $\{S_1, S_2, \ldots, S_n\}$ be the set of all embeddings of S in the exact graph $((V(G), E(G)), \Sigma(G), L(G))$, i.e. the exact graph obtained by removing the uncertainties from G, where $\Sigma(G)$ denotes the set of labels of G, and $L(G)$ denotes the labeling function of G. Let the edge set of each embedding S_i be $E(S_i) = \{e_{i_1}, e_{i_2}, \ldots, e_{i_{|E(S)|}}\}$, where subscript $i_j \in \{1, 2, \ldots, |E(G)|\}$. Note that all embeddings have the same number of edges, $|E(S)|$. The DNF counting problem is constructed as follows.

Step 1. For each edge e_j in the embeddings, create a boolean variable x_j. The probability, $\Pr(x_j)$, of x_j being assigned true is equal to the existence possibility, $P(e_j)$, of edge e_j.

Step 2. For each embedding S_i, construct a conjunctive clause $C_i = x_{i_1} \wedge x_{i_2} \wedge \cdots \wedge x_{i_{|E(S)|}}$, where x_{i_j} is the boolean variable created for edge $e_{i_j} \in E(S_i)$ in step 1.

Step 3. The output DNF formula F is the disjunction of all conjunctive clauses constructed for all n embeddings in step 2, i.e. $F = (x_{1_1} \wedge x_{1_2} \wedge \cdots \wedge x_{1_{|E(S)|}}) \vee \cdots \vee (x_{n_1} \wedge x_{n_2} \wedge \cdots \wedge x_{n_{|E(S)|}})$.

[2]See Definition 2. Note that the number of all embeddings of S in G is no larger than the number of all subgraph isomorphisms from S to G since two distinct subgraph isomorphisms may map S to the same subgraph in G.

The construction can be done in $\Theta(n|E(S)|)$ time using a hash table to store the variable created for each edge, where n is the number of embeddings of S in G, and $|E(S)|$ is the number of edges in S. It is easy to prove that $P(S \sqsubseteq_U G)$ is equal to the probability of F being satisfied by a randomly and independently chosen truth assignment to the variables in F, denoted as $\Pr(F)$. Thus, the problem of computing $P(S \sqsubseteq_U G)$ is transformed to the problem of computing $\Pr(F)$.

Example 4. *Consider uncertain graph G_1 and subgraph pattern S in Figure 1. S has 4 embeddings in G_1 as illustrated by the dotted circles in Figure 7. Four variables x_1, x_2, x_3, x_4 are created for edges (v_1, v_2), (v_1, v_3), (v_1, v_4) and (v_1, v_5) in the embeddings, respectively. The probabilities of x_1, x_2, x_3, x_4 being assigned true are* $\Pr(x_1) = 0.5$, $\Pr(x_2) = 0.6$, $\Pr(x_3) = 0.7$ *and* $\Pr(x_4) = 0.8$, *respectively. The clauses constructed for the embeddings are shown in Figure 7. Thus, the constructed DNF formula is $F = (x_1 \wedge x_2) \vee (x_1 \wedge x_4) \vee (x_3 \wedge x_4) \vee (x_2 \wedge x_3)$.*

Note that if F can be divided into several DNF subformulas $F_1, F_2, \ldots, F_k$ such that $F = F_1 \vee F_2 \vee \cdots \vee F_k$ and that F_i and F_j don't contain any common variables for $i \neq j$, then we can first compute $\Pr(F_i)$ for each subformula F_i and then compute $\Pr(F)$ by $\Pr(F) = 1 - \prod_{i=1}^{k}(1 - \Pr(F_i))$. Without loss of generality, we assume F is indivisible in the following discussion.

Based on this technique, an exact algorithm and an approximation algorithm are developed to compute $P(S \sqsubseteq_U G)$ in sequel.

5.2.2 Exact Algorithm

To compute $P(S \sqsubseteq_U G)$ exactly, we first construct a DNF formula $F = C_1 \vee C_2 \vee \cdots \vee C_n$ using the method given previously. By the *Inclusive-Exclusive Principle* [17], we have

$$\Pr(F) = \sum_{1 \le i \le n} \Pr(C_i) - \sum_{1 \le i < j \le n} \Pr(C_i \wedge C_j) + \cdots + (-1)^{n-1} \sum_{1 \le i_1 < i_2 < \cdots < i_n \le n} \Pr(C_{i_1} \wedge C_{i_2} \wedge \cdots \wedge C_{i_n}), \tag{6}$$

where $\Pr(C_{i_1} \wedge C_{i_2} \wedge \cdots \wedge C_{i_k})$ denotes the probability of $C_{i_1} \wedge C_{i_2} \wedge \cdots \wedge C_{i_k}$ being satisfied. Since each clause C_i in F is a conjunction of unengaged variables, we have

$$\Pr(C_{i_1} \wedge C_{i_2} \wedge \cdots \wedge C_{i_j}) = \prod_{x} \Pr(x), \tag{7}$$

where x is over all variables in $C_{i_1} \wedge C_{i_2} \wedge \cdots \wedge C_{i_j}$.

Example 5. *Also consider the example illustrated in Figure 7. We have* $\Pr(F) = 0.782$ *by Eq. (6).*

It takes $\Theta(k|E(S)|)$ time to compute $\Pr(C_{i_1} \wedge C_{i_2} \wedge \cdots \wedge C_{i_k})$, so $\Pr(F)$ can be computed by Eq. (6) in $\Theta(\sum_{k=1}^{n} \binom{n}{k} k|E(S)|) = \Theta(2^{n-1} n|E(S)|)$ time, where n is the number of embeddings of S in G, and $|E(S)|$ is the number of edges in S. Thus, the time complexity of the exact algorithm is $\Theta(2^{n-1} n|E(S)|)$.

5.2.3 *Approximation Algorithm*

The time complexity of the exact algorithm is exponential, so it can't scale for more than 30 embeddings. When S has a large number of embeddings in G, we propose an approximation algorithm to approximate $P(S \sqsubseteq_U G)$ by an interval efficiently. The algorithm consists of two steps.

Step 1. Transform the problem of computing $P(S \sqsubseteq_U G)$ to the DNF counting problem by constructing a DNF formula F as presented previously.

Step 2. Approximate the satisfaction probability $\Pr(F)$ in polynomial time using an interval $[l, u]$ such that the width of $[l, u]$ is at most $\varepsilon \cdot minsup$ and that $\Pr(F) \in [l, u]$.

Note that the width of the interval $[l, u]$ is required to be at most $\varepsilon \cdot minsup$ in step 2. This is due to the following reason. For an uncertain graph database $D = \{G_1, G_2, \ldots, G_n\}$ and a subgraph pattern S in D, let $[l_i, u_i]$ be an approximated interval of $P(S \sqsubseteq_U G_i)$ such that $P(S \sqsubseteq_U G_i) \in [l_i, u_i]$ and $|u_i - l_i| \le \varepsilon \cdot minsup$ for $1 \le i \le n$, and let $\bar{l} = \frac{1}{n}\sum_{i=1}^{n} l_i$ and $\bar{u} = \frac{1}{n}\sum_{i=1}^{n} u_i$. We have that the expected support, $esup_D(S)$, of S in D must be contained in the interval $[\bar{l}, \bar{u}]$ and $|\bar{u} - \bar{l}| \le \varepsilon \cdot minsup$. Hence, when determining whether S can be output or not using the approximated interval $[\bar{l}, \bar{u}]$ of $esup_D(S)$, either condition 1 or condition 2 in Figure 5 will be satisfied, so S can be determined whether to be output or not efficiently.

A number of algorithms [11,16] have been proposed to compute the interval $[l, u]$ in step 2. Although the deterministic approximation algorithms such as [16] can produce the desired intervals that certainly enclose the expected support, all these algorithms have too high time complexity to be applicable in practice. For this reason, we use the fully polynomial randomized approximation scheme (FPRAS) proposed by Karp and Luby [11] to achieve both high accuracy and high efficiency. For a given DNF formula F, an absolute error ϵ and a real number $\delta \in [0, 1]$, the FPRAS can find an interval $[l, u]$ such that $\Pr(F) \in [l, u]$ and $|u - l| \le \epsilon$ with probability $1 - \delta$ in polynomial time.

Algorithm 1 illustrates the proposed approximation algorithm, called *Approx-Exp-Sup*. Line 1 constructs the DNF formula F. Line 2 set the absolute error $\varepsilon' = \varepsilon \cdot minsup/2$. Lines 3–12 are the FPRAS, which returns an estimate $\hat{p}$ of $\Pr(F)$ such that $|\hat{p} - \Pr(F)| \le \varepsilon'$ with probability $1 - \delta$. Line 13 returns interval $[\hat{p} - \varepsilon', \hat{p} + \varepsilon']$ to approximate $\Pr(F)$. It is evident that the returned interval has width $2\varepsilon' = \varepsilon \cdot minsup$ and $\Pr(F)$ is contained in $[\hat{p} - \varepsilon', \hat{p} + \varepsilon']$ with probability $1 - \delta$.

The time complexity of the approximation algorithm is analyzed as follows. The construction at line 1 can be done in $\Theta(n|E(S)|)$ time. Line 2 computes Z in $\Theta(n|E(S)|)$ time. Lines 7–11 loop for N times. For each loop, line 8 spends $O(n|E(S)|)$ time to generate a random assignment, and the condition at line 10 can be tested in $O(i|E(S)|)$ time. Since i is uniformly randomly picked out from $\{1, 2, \ldots, n\}$, line 10 can be tested expectedly in $O(n|E(S)|/2)$ time. Thus, the expected time complexity of the approximation algorithm is $O(Nn|E(S)|)$, where $N = \frac{4n\ln(2/\delta)}{\varepsilon'^2} = \frac{16n\ln(2/\delta)}{\varepsilon^2 minsup^2}$ is the number of samplings, i.e. loops, carried out by the FPRAS.

5.2.4 *Trade-off between Exact Algorithm and Approximation Algorithm*

Here, we discuss how to adaptively decide which algorithm should be used to compute $P(S \sqsubseteq_U G)$ for an input uncertain graph G and an input subgraph pattern S. As analyzed previously, the time complexity of the exact algorithm is $T_{exact} = \Theta(2^{n-1}n|E(S)|)$, and the time complexity of the approximation algorithm is T_{approx}

Algorithm 1: The *Approx-Occ-Prob* procedure.

```
Input: a subgraph pattern $S$, an uncertain graph $G$, the set,
       $\{S_1, S_2, \ldots, S_n\}$, of all embeddings of $S$ in $G$, a
       threshold $minsup \in [0,1]$, a relative error tolerance
       $\varepsilon \in [0,1]$ and a real number $\delta \in [0,1]$.
Output: an interval that approximates the probability of $S$
        occurring in $G$.
1.  Construct the DNF formula $F = C_1 \vee C_2 \vee \cdots \vee C_n$;
2.  $\varepsilon' \leftarrow \varepsilon \cdot minsup/2$;
    /* Estimate $\Pr(F)$ by the FPRAS [11] using $\varepsilon'$ and $\delta$. */
3.  $N \leftarrow 4n\ln(2/\delta)/\varepsilon'^2$;
4.  $Z \leftarrow \Pr(C_1) + \Pr(C_2) + \cdots + \Pr(C_n)$;
5.  $X \leftarrow Y \leftarrow 0$;
6.  for $loop \leftarrow 1$ to $N$ do
7.      $i \leftarrow$ a uniform random integer in $\{1, 2, \ldots, n\}$;
8.      Randomly choose a truth assignment $\pi$ satisfying $C_i$;
9.      $Y \leftarrow Y + \Pr(\pi)$;
10.     if $\pi$ doesn't satisfy $C_j$ for all $1 \le j \le i$ then
11.         $X \leftarrow X + \Pr(\pi)$;
12. $\hat{p} \leftarrow XZ/Y$;
13. return $[\hat{p} - \varepsilon', \hat{p} + \varepsilon']$;
```

$= O(\frac{16n^2\ln(2/\delta)|E(S)|}{\varepsilon^2 minsup^2})$. Therefore, if $T_{exact} \ge T_{approx}$, that is,

$$\frac{2^{n-5}}{n} \ge \frac{\ln(2/\delta)}{\varepsilon^2 minsup^2},$$

then we choose the approximation algorithm to compute $P(S \sqsubseteq_U G)$, otherwise we use the exact algorithm.

Example 6. *Consider uncertain graph G_1 and subgraph pattern S in Figure 1. The number of edges in S is $|E(S)| = 2$. As shown in Figure 7, S has 4 embeddings in G, i.e. $n = 4$. If we use the exact algorithm to compute $P(S \sqsubseteq_U G)$, the number of operations need to be done is $\Theta(2^{n-1}n|E(S)|) = \Theta(64)$. Supposing that $minsup = 0.1$, $\varepsilon = 0.1$ and $\delta = 0.1$, the approximation algorithm need to do $O(\frac{16n^2\ln(2/\delta)|E(S)|}{\varepsilon^2 minsup^2}) = O(15338150)$ operations to approximate $P(S \sqsubseteq_U G)$. In this case, the exact algorithm is much more efficient than the approximation algorithm.*

5.2.5 *Algorithm for Computing Expected Supports*

By integrating the exact algorithm and the approximation algorithm proposed above, we develop the *Approx-Exp-Sup* algorithm to compute the expected support of a subgraph pattern as shown in Algorithm 2. Initially, the interval $[l, u]$ is set to $[0, 0]$ at line 1. Then, for each uncertain graph $G_i \in D$, we first determine whether to use the exact algorithm or the approximation algorithm to compute $P(S \sqsubseteq_U G_i)$ at line 3. Let $[\alpha, \beta]$ be the resulting approximated interval of $P(S \sqsubseteq_U G_i)$ produced by the selected algorithm. Specificially, $\alpha = \beta = P(S \sqsubseteq_U G_i)$ for the exact algorithm. Then, α and β are added to l and u at lines 8 and 9, respectively. Finally, $[l/n, u/n]$ is output at line 10 as the approximated interval of $esup_D(S)$. It is easy to prove that $l/n \le esup_D(S) \le u/n$ and $|u/n - l/n| \le \varepsilon \cdot minsup$.

5.3 Complete Algorithm

The complete algorithm, called *MUSE* (Mining Uncertain Subgraph pattErns), is outlined in Algorithm 3. The input of *MUSE* is an uncertain graph database D, a threshold $minsup \in [0, 1]$, a relative error tolerance $\varepsilon \in [0, 1]$ and a real number $\delta \in [0, 1]$. The output of *MUSE* is an approximate set of frequent subgraph patterns in D. The algorithm works as follows.

Algorithm 2: The *Approx-Exp-Sup* procedure.

```
Input: a subgraph pattern $S$, an uncertain graph database
       $D = \{G_1, G_2, \ldots, G_n\}$, a threshold $minsup$, a
       relative error tolerance $\varepsilon$, a real number $\delta$ and the set
       $X_i$ of all embeddings of $S$ in $G_i$ for $1 \leq i \leq n$.
Output: an interval approximating $esup_D(S)$.
1.  $l \leftarrow u \leftarrow 0$;
2.  for $i \leftarrow 1$ to $n$ do
3.      if $2^{|X_i|-5}/|X_i| \geq \ln(2/\delta)/(\varepsilon \cdot minsup)^2$ then
4.          $[\alpha, \beta] \leftarrow$ Approx-Occ-Prob($S$, $G_i$, $X_i$, $minsup$, $\varepsilon$, $\delta$);
5.      else
6.          Construct the DNF formula $F$ based on $X_i$;
7.          $\alpha \leftarrow \beta \leftarrow \Pr(F)$ computed by Eq. (6);
8.      $l \leftarrow l + \alpha$;
9.      $u \leftarrow u + \beta$;
10. return $[l/n, u/n]$;
```

First, initialize the result set F to be empty at line 1. Then, scan the edges of the uncertain graphs in D to obtain all subgraph patterns with one edge and push them into an empty stack T at line 2. Next, perform depth-first search on the search tree of subgraph patterns to discover an approximate set F of frequent subgraph patterns at lines 3 to 14. Finally, line 15 outputs F as an answer.

The depth-first search on the search tree is performed as follows. While T is not empty, run the following steps.

Step 1. Pop the subgraph pattern S on the top of T at line 4. For each uncertain graph $G_i \in D$, find the subgraph isomorphisms from S to G_i at line 6 and get the set X_i of all embeddings of S in G_i under the subgraph isomorphisms from S to G_i at line 7. The subgraph isomorphism problem has been extensively studied. Here, we take advantage of the depth-first search to find the subgraph isomorphisms from S to G_i incrementally based on the subgraph isomorphisms from its parent in the search tree to G_i. Our method is briefly introduced in Appendix A.

Step 2. Call the *Approx-Exp-Sup* procedure to approximate the expected support of S in D by an interval $[l, u]$ at line 8. If $u < minsup$, i.e. condition 2 in Figure 5 is satisfied, then S will not be output, and the following step 3 will be skipped. By skipping step 3, all descendants of S will not be examined, i.e. they are pruned.

Step 3. If $l \geq (1 - \varepsilon)minsup$ and $u \geq minsup$, i.e. condition 1 in Figure 5 is satisfied, then add S to F at line 10 and scan the edges incident on the vertices of the embeddings of S in the uncertain graphs to obtain all direct supergraphs of S at line 11. For each direct superpattern S' of S, if S is the parent of S' in the search tree, then push S' into stack T at line 14, otherwise S' is a child of another subgraph pattern S'' and should not be examined in the subtree rooted at S. Note that Parent(S') on line 13 returns the parent of S' in the search tree. The detailed procedure of function Parent depends on the scheme used to build the search tree. For example, if the scheme in [27] is used, then Parent(S') returns the subgraph pattern with its minimum DFS code [27] being the longest prefix of the minimum DFS code of S'.

6 EXPERIMENTS

The *MUSE* algorithm was implemented in C, and experiments were performed to evaluate the efficiency, approximation quality and scalability of *MUSE*, and the impact of uncertainties on the efficiency of *MSUE*. In our implementation, we use the DFS coding scheme proposed in [27] to construct search trees. All experiments were performed on an IBM ThinkPad T61 notebook with 2GHz CPU and 2GB RAM, running Windows XP.

Algorithm 3: The *MUSE* algorithm.

```
Input: an uncertain graph database $D = \{G_1, G_2, \ldots, G_n\}$, a
       threshold $minsup \in [0, 1]$, a relative error tolerance
       $\varepsilon \in [0, 1]$ and a real number $\delta \in [0, 1]$.
Output: an approximate set of frequent subgraph patterns in
        $D$.
1.  $F \leftarrow \emptyset$;
2.  $T \leftarrow$ {all subgraph patterns in $D$ with one edge};
3.  while $T \neq \emptyset$ do
4.      $S \leftarrow$ Pop($T$);
5.      for $i \leftarrow 1$ to $n$ do
6.          Find the subgraph isomorphisms from $S$ to $G_i$;
7.          $X_i \leftarrow$ {all embeddings of $S$ in $G_i$};
8.      $[l, u] \leftarrow$ Approx-Exp-Sup($S$, $D$, $minsup$, $\varepsilon$, $\delta$,
        $X_1, X_2, \ldots, X_n$);
9.      if $l \geq (1 - \varepsilon)minsup$ and $u \geq minsup$ then
10.         $F \leftarrow F \cup \{S\}$;
11.         $Y \leftarrow$ {all direct superpatterns of $S$};
12.         foreach $S' \in Y$ do
13.             if Parent($S'$) = $S$ then
14.                 Push($S'$, $T$);
15. return $F$;
```

We experimented using a real uncertain graph database. The real uncertain graph database was obtained from the STRING database[3]. It contains the PPI networks of six organisms, which are summarized in Table 1. In Table 1, $|V|$ indicates the number of vertices, $|E|$ indicates the number of edges, and Avg(P) indicates the average value of existence possibilities of edges. Moreover, all vertices are labeled with COG protein functions[4].

Table 1: Summary of the real uncertain graph database.

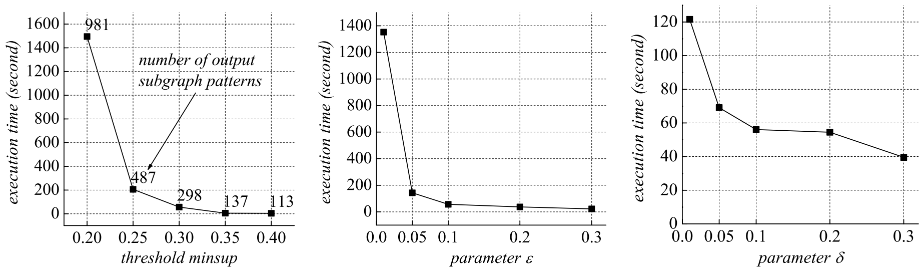
organism	$\|V\|$	$\|E\|$	Avg(P)
fission yeast	162	300	0.148
fruit fly	3751	7384	0.456
house mouse	199	286	0.413
rat	130	178	0.374
thale cress	513	1168	0.444
worm	514	960	0.190

6.1 Time Efficiency of *MUSE*

We first investigated the time efficiency of *MUSE* on the real uncertain graph database with respect to the threshold $minsup$ and the parameters ε and δ. Figure 8(a) shows the execution time of *MUSE* while $minsup$ varies from 0.2 to 0.4, $\varepsilon = 0.1$ and $\delta = 0.1$. The execution time decreases substantially while $minsup$ increases. This is because the number of output frequent subgraph patterns decreases rapidly as $minsup$ becomes larger. Figure 8(b) shows the execution time of *MUSE* while ε varies from 0.01 to 0.3, $minsup = 0.3$ and $\delta = 0.1$. The execution time decreases rapidly while ε increases. The reason is that the time spent by the *Approx-Occ-Prob* procedure decreases quadratic to the increase of ε as analyzed in Section 5.2.3. Figure 8(c) shows the execution time of *MUSE* while δ varies from 0.01 to 0.3, $minsup = 0.3$ and $\varepsilon = 0.1$. The execution time decreases rapidly while δ increases. This is because the time complexity of the *Approx-Occ-Prob* procedure is proportional to $\ln(2/\delta)$ as analyzed in Section 5.2.3.

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

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

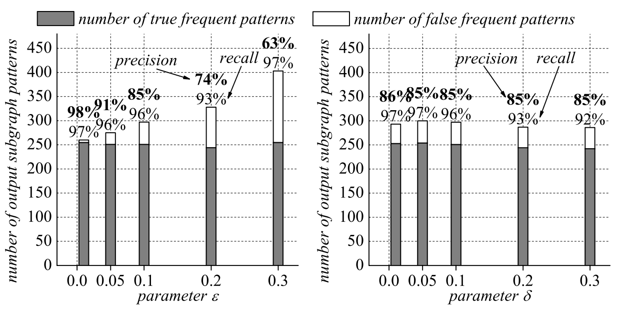


(a) Varying $minsup$ (b) Varying ε (c) Varying δ

Figure 8: Execution time of *MUSE* with respect to threshold $minsup$ and parameters ε and δ.

6.2 Approximation Quality of *MUSE*

Since *MUSE* is an approximate mining algorithm, we evaluated its approximation quality with respect to ε and δ on the real uncertain graph database. The approximation quality is measured by the *precision* and *recall* metrics. Precision is the percentage of true frequent subgraph patterns in the output subgraph patterns. Recall is the percentage of returned subgraph patterns in the true frequent subgraph patterns. Since it is NP-hard to find all true frequent subgraph patterns, we regarded the subgraph patterns discovered using $\varepsilon = 0.01$ and $\delta = 0.01$ as the true frequent subgraph patterns. Figure 9(a) shows the details of the output subgraph patterns while ε varies from 0.01 to 0.3, $\delta = 0.1$ and $minsup = 0.3$. Each percentage above in the figure indicates the precision, and each percentage below indicates the recall. We can see that the precision of *MUSE* decreases and the recall remains stable while ε increases. This is because (1) when ε becomes larger, more false frequent subgraph patterns will be returned, so reducing the precision; (2) when δ is fixed, the probability of a frequent subgraph pattern being returned is also fixed, thus the number of output true frequent subgraph patterns don't change significantly. Figure 9(b) shows the experimental results while δ varies from 0.01 to 0.3, $\varepsilon = 0.1$ and $minsup = 0.3$. The precision remains stable but the recall decreases while δ increases. The reason is that (1) the fixed ε determines the expected number of false frequent subgraph patterns to be returned, so the precision remains stable; (2) while δ increases, the probability of a frequent subgraph pattern being output decreases, thus the number of returned true frequent subgraph patterns decreases, reducing the recall. All the experimental results verify that *MUSE* can have very high approximation quality.

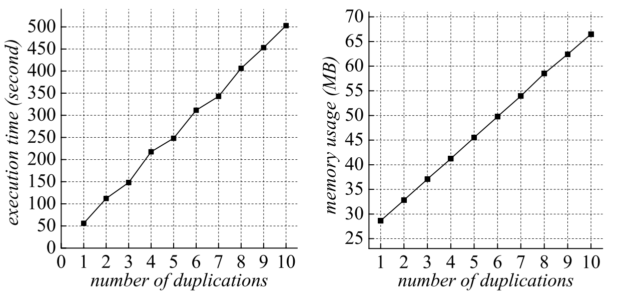


(a) Varying ε (b) Varying δ

Figure 9: Approximation quality of *MUSE* with respect to parameters ε and δ.

6.3 Scalability of *MUSE*

We also examined the scalability of *MUSE* with respect to the number of uncertain graphs in an uncertain graph database. We controlled the number of uncertain graphs by duplicating the uncertain graphs in the database. Figure 10 shows the execution time and the memory usage of *MUSE* on the duplicated real uncertain graph database while the number of duplications varies from 1 to 10, $minsup = 0.3$, $\varepsilon = 0.1$ and $\delta = 0.1$. Both the execution time and the memory usage increase linearly to the increasing of the number of uncertain graphs. The experimental results verify that *MUSE* is very scalable to large uncertain graph databases.



(a) Execution time (b) Memory usage

Figure 10: Scalability of *MUSE* with respect to the number of uncertain graphs.

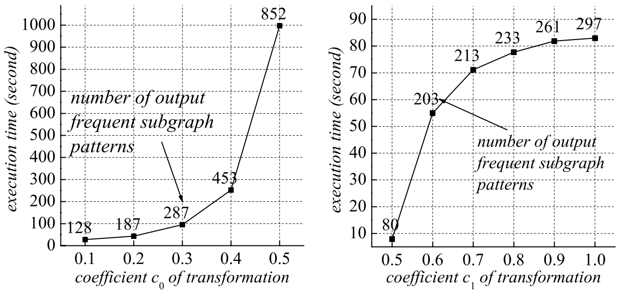
6.4 Impact of Uncertainties on *MUSE*

This experiment investigated the impact of distributions of uncertainties on the efficiency of *MUSE*. To vary the distribution of uncertainties, we imposed mathematical transformations to the uncertainties of each uncertain graph. The transformation is of the form

$$f(x) = \begin{cases} 1 & \text{if } c_1x + c_0 > 1, \\ 0 & \text{if } c_1x + c_0 < 0, \\ c_1x + c_0 & \text{otherwise,} \end{cases}$$

where $c_0, c_1 \in [0, 1]$. It transforms the existence possibility value $x \in [0, 1]$ of an edge to $f(x) \in [0, 1]$.

We ran *MUSE* with $minsup = 0.3$, $\varepsilon = 0.1$ and $\delta = 0.1$ on the transformed real uncertain graph databases. Figure 11(a) shows the execution time of *MUSE* while the coefficient c_0 of the transformation varies from 0 to 0.5, and the coefficient $c_1 = 0.5$. Each integer on the line indicates the number of output subgraph patterns. We can see that the execution time increases as c_0 becomes larger. This is because the larger c_0 leads to the increase in the existence possibilities of edges, thus increasing the expected supports of all subgraph patterns. Since $minsup$ is fixed, more subgraph patterns will be output as frequent subgraph patterns, so increasing the execution time. Figure 11(b) shows the execution time of *MUSE* while the coefficient c_1 of the transformation varies from 0.5 to 1 and $c_0 = (1 - c_1)\mu$, where μ is the mean value of the existence possibilities of the edges in the uncertain graph to be transformed. It is easy to show that the mean value of the existence possibilities after transformation is also μ. The execution time increases as c_1 becomes larger. This is because with the increasing of c_1, the variance of the existence possibilities becomes larger, and more edges will have high existence possibilities. It increases the number of subgraph patterns with high expected supports, thus increasing the execution time consequently.



(a) Varying c_0 (b) Varying c_1

Figure 11: Impact of uncertainties on the efficiency of *MUSE*.

7 CONCLUSIONS

This paper investigates the problem of mining frequent subgraph patterns on uncertain graph data. The frequent subgraph pattern mining problem is formalized by introducing the expected support measure. An approximate mining algorithm, called *MUSE*, is proposed to discover an approximate set of frequent subgraph patterns from an uncertain graph database. The analysis and the experimental results show that *MUSE* has high efficiency, high approximation quality and high scalability.

8 ACKNOWLEDGMENTS

This research work was supported in part 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 Key Program of the NSF of China under Grant No. 60533110.

9 References

[1] C. C. Aggarwal, Y. Li, J. Wang, and J. Wang. Frequent pattern mining with uncertain data. In *KDD*, pages 29–38, 2009.

[2] S. Asthana, O. D. King, F. D. Gibbons, and F. P. Roth. Predicting protein complex membership using probabilistic network reliability. *Genome Research*, 14(6):1170–1175, 2004.

[3] 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.

[4] G. Cormode and A. McGregor. Approximation algorithms for clustering uncertain data. In *PODS*, pages 191–200, 2008.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[6] J. Ghosh, H. Q. Ngo, S. Yoon, and C. Qiao. On a routing problem within probabilistic graphs and its application to intermittently connected networks. In *INFOCOM*, pages 1721–1729, 2007.

[7] P. Hintsanen and H. Toivonen. Finding reliable subgraphs from large probabilistic graphs. *Data Min. Knowl. Discov.*, 17(1):3–23, 2008.

[8] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *ICDM*, page 549, 2003.

[9] J. Huan, W. Wang, J. Prins, and J. Yang. Spin: mining maximal frequent subgraphs from graph databases. In *KDD*, pages 581–586, 2004.

[10] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD*, pages 13–23, 2000.

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

[12] Y. Ke, J. Cheng, and W. Ng. Correlation search in graph databases. In *KDD*, pages 390–399, 2007.

[13] M. Koyutürk, A. Grama, and W. Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. *Bioinformatics*, 20(Suppl. 1):i200–i207, 2004.

[14] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM*, pages 313–320, 2001.

[15] Y. Liu, J. Li, and H. Gao. Summarizing graph patterns. In *ICDE*, pages 903–912, 2008.

[16] M. Luby and B. Velickovic. On deterministic approximation of dnf. In *STOC*, pages 430–438, 1991.

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

[18] S. Nijssen and J. N. Kok. A quickstart in frequent structure mining can make a difference. In *KDD*, pages 647–652, 2004.

[19] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *KDD*, pages 228–238, 2005.

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

[21] S. Tsang, B. Kao, K. Y. Yip, W.-S. Ho, and S. D. Lee. Decision trees for uncertain data. In *ICDE*, pages 441–444, 2009.

[22] L. G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[23] N. Vanetik. Discovering frequent graph patterns using disjoint paths. *TKDE*, 18(11):1441–1456, 2006.

[24] C. Wang, W. Wang, J. Pei, Y. Zhu, and B. Shi. Scalable mining of large disk-based graph databases. In *KDD*, pages 316–325, 2004.

[25] J. Wang, Z. Zeng, and L. Zhou. Clan: An algorithm for mining closed cliques from large dense graph databases. In *ICDE*, page 73, 2006.

[26] X. Yan, H. Cheng, J. Han, and P. S. Yu. Mining significant graph patterns by leap search. In *SIGMOD*, pages 433–444, 2008.

[27] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *ICDM*, page 721, 2002.

[28] X. Yan and J. Han. Closegraph: mining closed frequent graph patterns. In *KDD*, pages 286–295, 2003.

[29] Z. Zeng, J. Wang, L. Zhou, and G. Karypis. Out-of-core coherent closed quasi-clique mining from large dense graph databases. *TODS*, 32(2):13, 2007.

[30] Q. Zhang, F. Li, and K. Yi. Finding frequent items in probabilistic data. In *SIGMOD*, 2008.

APPENDIX

A INCREMENTAL METHOD FOR FINDING SUBGRAPH ISOMORPHISMS

We briefly introduce our method for finding subgraph isomorphisms from a subgraph pattern S to a graph G. If S contains only one edge, then we trivially scan all edges of G to find the subgraph isomorphisms. If S consists of more than one edge, we can find the subgraph isomorphisms in an incremental manner. Let S' be the parent subgraph pattern of S in the search tree, and let (u, v) be the only edge in $E(S) \setminus E(S')$. Note that a subgraph isomorphism from S to G must contains a subgraph isomorphism from S' to G. Thus, we can find the subgraph isomorphisms from S to G incrementally based on the subgraph isomorphisms from S' to G.

Suppose both u and v are contained in S'. For every subgraph isomorphism f' from S' to G, if edge $(f'(u), f'(v))$ is contained in $E(G)$ and the label of (u, v) is identical to the label of $(f'(u), f'(v))$, then f' is also a subgraph isomorphism from S to G.

Suppose $u \in V(S')$ but $v \notin V(S')$. For every subgraph isomorphism f' from S' to G, if there exists an edge $(f'(u), w)$ in G such that the label of (u, v) is identical to the label of $(f'(u), w)$ and that $w \neq f'(x)$ for all vertices $x \in V(G)$, then $f' \cup \{u \rightarrow f'(u), v \rightarrow w\}$ is a subgraph isomorphism from S to G.

It is obvious that our method is more efficient than the methods that find subgraph isomorphisms from scratch.