Frequent Subgraph Mining in Uncertain Graphs
An Exploratory Analysis

Edouard Fouché
March 10, 2015

Karlsruhe Institute of Technology (KIT)

Abstract. Uncertainties are inherent in graph data and in many applications. Mining frequent subgraphs has been extensively studied in certain graphs, but transferring the problem to uncertain data is not trivial. This field of research is relatively young and to the best of our knowledge, only two papers [1] and [2] have been released about this topic so far. Both papers were written by Zou et al. and describe two different approaches for finding frequent patterns in uncertain graph databases. The first approach solves the problem using expected semantics, the second one using probabilistic semantics. Both have been proven to be efficient and accurate for large uncertain graph databases, but there is a lack of explanation about how these two approaches differ in terms of mining result and efficiency. However, it has been reported that expected semantics are better to use as an exploratory tool and structural pattern detection, whereas probabilistic semantics as a mining tool for extracting features: classifying, clustering, frequent pattern mining. In this paper, after introducing the topic, we will focus on these two approaches and try to emphasize, in a comparative study, the strength and drawbacks of each of them. We will expose the concepts required to understand the problems inherent in mining uncertain data. Finally, we will review and compare associated analytical and experimental results.

1 INTRODUCTION

1.1 Motivations

Graphs are common data structures in the field of computer science and can be used to represent complicated relationships between entities. Graphs have been widely used in bioinformatics, network modeling, spatial databases, etc. Because of the expressiveness of this data structure, various intelligent tools and concepts have been developed to analyze and understand them. One of the most powerful tool for detecting recurring substructures, that has been introduced recently, is Frequent subgraph mining [3]. It represents the problem as follows: Given a set $D$ of graphs and a number $0 < \varphi < 1$, find all subgraphs that occur in at least $\varphi:|D|$ graphs in $D$. The percentage of graphs in $D$ that contain a subgraph $S$ is called support.

However, a number of applications of graph data are subject to uncertainties, which are mostly due to incompleteness, errors and imprecisions in measuring methods. Sometimes, it is also more relevant to represent data by a probability distribution rather than discrete values. This affirmation holds particularly for sensor measures. There are few studies so far that investigate this problem regarding this parameter, and a lot of assumptions, that have been made in certain data, do not hold because of uncertainties. For example, support counting does not make sense anymore, because vertices and edges are uncertain. This measure was redefined in [1] as expected support under the expected semantics. Besides, a new measure, so called $\varphi$ – frequent probability, was introduced in [2] for mining under probabilistic semantics. The problem of mining patterns in uncertain graph data has been proven to be NP-hard, and is computationally more challenging than mining subgraphs in certain graphs. We will discuss this fact in this paper and show how these two approaches attempt to solve this problem.
1.2 Applications

Various applications have to deal with uncertainties. Two major examples of applications are discussed in the literature in which uncertainties are inherent: Protein-protein interaction (PPI) networks [10] and routing protocols in wireless networks [11].

In bioinformatics, PPI networks are graphs that represent interactions between proteins. Proteins are represented by vertices and interactions are represented by edges in between. There exists a variety of methods for the detection of PPI networks. However, it has been reported that all methods produce noise, detect interactions that do not exist and miss also a fraction of the real interactions. This is why it is more appropriate to represent a PPI network as an uncertain graph, where edges and vertices can be associated with a probability distribution to represent their probability to exist in reality. In this example, mining methods for uncertain data could help predict the membership of a protein in a partially known protein complex and therefore, support discoveries in the field of biology and the finding of new drugs. This area has been studied by [10].

[11] shows how wireless networks can be modeled as an uncertain graph. Vertices represent devices in the network, called “nodes”, and edges the probability that the link is working normally between two endpoints. Detecting frequent subpattern could help the design of protocols for routing in wireless networks, so that they further jumping to node whose link has the highest confidence to work and detect paths in the network best suited to transmit data.

In mining uncertain data, discovered knowledge is generally associated with a confidence value, indicating the possibility of the knowledge existing in reality. Thus, only results occurring with a high confidence can be considered as useful.

1.3 Related Work

A substantial quantity of algorithms have been designed to extract knowledge from exact graph data. Most of the work focuses on extracting subgraph patterns, such as [3, 8, 9] or reducing redundant subgraphs. Some variants of the frequent subgraph pattern mining have also been studied, such as the discovery of frequent quasi-cliques or correlated subgraph patterns [4]. However, these algorithms cannot be used for mining uncertain data, because they cannot handle uncertainties inherent in graph data.

Mining structured data in uncertain relational data models has also been studied [6, 7]. Existing work has focused on typical operations in the context of data mining, such as clustering, frequent itemset mining and classification. However, this work cannot be shifted to uncertain graph data, because uncertainties are not of the same nature.

More recently, two papers focusing on mining uncertain graph data have been written by Zou et al. [1,2] and investigate this topic under expected and probabilistic semantics and form the core of our study.

2 FREQUENT SUBGRAPH MINING UNDER EXPECTED SEMANTICS

Frequent subgraph mining in uncertain graph data has been studied at first by Zou et al. in [1]. They proposed to define a new measure called expected support, in order to evaluate the significance of subgraphs. Expected support quantifies the expected value of the support of a subgraph S in all implicated graph databases. We are going to expose more details about this approach in the following subsections.

2.1 Model of Uncertain Graphs

Before formally introducing the model of uncertain graphs, let us approach the model with an example. This example was used in the introduction of [1]. Let us consider an uncertain graph database D, which contains
two uncertain graphs, $G_1$ and $G_2$. This uncertain graph database is illustrated in Fig. 1. Each vertex and edge is labelled with a text ($A$ or $B$ for vertices and $x$ or $y$ for edges) and each edge is associated with a real number between 0 and 1, an existence possibility, representing their possibility of existing between the endpoints in reality, i.e. in an exact graph instance. $G_1$ can be viewed as a representation of 16 exact graphs, called *implicated graphs*, whose probability distribution can be derived from the existence possibilities of the edges. In the same way, $G_2$ represents 8 implicated graphs. An exact graph database is obtained by combining any implicated graphs from $G_1$ with any implicated graphs from $G_2$. Thus, due to various combinations of implicated graphs, $D$ has in total $16 \times 8 = 128$ implicated graphs databases. Then, we can model the significance of a certain subgraph pattern by considering both the number of occurrence of the pattern over all implicated graph databases and the existence probability of the implicated graph databases in which the pattern occurs.

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

The *existence possibility*, $P((u,v))$, of an edge $(u,v)$ is expressed by the probability that the edge between $u$ and $v$ exists in reality. For example, if $P(u,v)=1$, then the edge between the vertices $u$ and $v$ always exists. Thus, an exact graph can be represented by an uncertain graph, whose existence possibility for all edges is equal to 1. In that sense, the model of an uncertain graph is just an extension of the existing commonly accepted definition of an exact graph and contains it. However, an uncertain graph *implicates* a set of exact graphs. 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 (1) $G$ and $I$ have the exact same ensemble of vertices, (2) the ensemble of edges $E'$ and labels $\Sigma$ of $I$ are included in the system of $G$, and (3) $L'$ is the function obtained by restricting $L$ to $V' \cup E'$.

To simplify the problem, it is assumed that all existence possibilities of edges are independent. This assumption has been shown to be reasonable in some real applications like PPI networks and routing protocols of wireless network. However, this may not be sufficient for some other existing or future applications and may be a fundamental weakness of this approach. Based on this assumption, the possibility of an uncertain graph $G$ implicating an exact graph $I$, denoted by $P(G \Rightarrow I)$ is equal to

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

(1)

where $P(e)$ is the existence possibility of the edge $e$. Let $Imp(G)$ denote the set of all implicated graphs of $G$. It is easy to show that $|Imp(G)| = 2^{|E(G)|}$ and that the function $P(G \Rightarrow I)$ defines a probability distribution over $Imp(G)$. This means that the number of implicated graphs from an uncertain graph increases exponentially with the number of edges in the uncertain graph.

An uncertain graph database is a set of uncertain graphs and represents a set of implicated graph databases. Let $D = \{G_1,G_2,\ldots,G_n\}$ be an uncertain graph database containing $n$ uncertain graphs. A
set of exact graphs \( d = \{I_1, I_2, \ldots, I_n\} \) is an implicated graph database of \( D \), if and only if if \( G_i \Rightarrow I_i \) for all \( i \in [1, n] \). Due to the various combinations between the set of implicated graphs of each uncertain graph \( G_i \), the set of all implicated graph databases of \( D \), denoted by \( \text{Imp}_{db}(D) \), is equal to \( \prod_{i=1}^{n} 2^{|E(G_i)|} \).

Similarly to the assumptions we have made about the independence between edges, it is assumed that the uncertain graphs in an uncertain graph database are mutually independent, which leads again to a simplification of the problem. However, it is not sure that this assumption is realistic, because objects in a database may depend from each other, e.g. when measuring chemical reactions, chains reactions may result in graphs that depend on each other, another example could be the design of a sensor network, in which the position of each sensor or group of sensors depends on the position of other sensors. The possibility of a graph database \( d \) being implicated by an uncertain graph database \( D \) is equal to

\[
P(D \Rightarrow d) = \prod_{i=1}^{n} P(G_i \Rightarrow I_i),
\]

where \( P(G_i \Rightarrow I_i) \) is the probability of \( G_i \) implicating \( I_i \), as defined in (1). Similarly, \( P(D \Rightarrow d) \) defines a probability distribution over \( \text{Imp}_{db}(D) \).

### 2.2 Problem Definition

In conventional subgraph pattern mining, the frequency of a subgraph pattern \( S \) in an exact graph database \( D \) is expressed by taking into account the number of occurrence of the subgraph among the different graphs \( G_i \) in \( D \). Such measure is called support, it corresponds to the relative frequency of occurrences of \( S \) in the database \( D \), and is defined as follows: \( \text{sup}_D(S) = \frac{|\{G \in D : S \subseteq_{ex} G \text{ and } G \in D\}|}{|D|} \), with \( S \subseteq_{ex} G \) meaning that \( S \) is subgraph isomorphic to \( G \).

A subgraph pattern \( S = ((V', E'), \Sigma', L') \) is subgraph isomorphic to an exact graph \( G = ((V, E), \Sigma, L) \), denoted by \( S \subseteq_{ex} G \), if there exists an injection \( f : V' \rightarrow V \) such that (1) \( L'(v) = L(f(v)) \) for any \( v \in V' \), (2) \( (f(u), f(v)) \in E \) for any \( (u, v) \in E' \), (3) \( L'((u, v)) = L((f(u), f(v))) \) for any \( (u, v) \in E \). The injection is called a subgraph isomorphism from \( S \) to \( G \). A subgraph is called subgraph pattern in an uncertain graph database \( D \) if and only if it is subgraph isomorphic to at least one of the uncertain graph in \( D \).

However, the traditional definition of the support does not hold in the context of uncertain graph data, because exact subgraphs are embedded in uncertain graphs only in a probabilistic sense. Thus, it doesn’t make sense to quantify the frequency of subgraphs, using this measure, under this new context. To address this issue, we may redefine support counting as expected support by using the traditional support definition with the probability distribution we introduced in the previous subsection. Since (2) represents a probability distribution over all sets of implicated graph database \( d \), denoted \( \text{Imp}_{db}(D) \), from an uncertain graph database \( D \), the value of the expected support of a subgraph \( S \) may be obtained by summing up the products between traditional support and existence possibilities among all elements of \( \text{Imp}_{db}(D) \). To put it in a nutshell, the expected support can be formalized as follows:

\[
\text{esup}_D(S) = \sum_{d \in \text{Imp}_{db}(D)} \text{sup}_d(S) \times P(D \Rightarrow d),
\]

Thus, a subgraph will be considered as frequent in an uncertain graph database \( D \) if the expected support of \( S \) is not less than a user-specified threshold \( \text{minsup} \in [0, 1] \). Then, the problem of discovering frequent subgraph patterns on an uncertain graph database can be defined as follows: Given an uncertain graph database \( D \) and a minimum expected support \( \text{minsup} \), find all frequent subgraph patterns in \( D \), i.e. whose expected support is not less than \( \text{minsup} \).

This is the definition of the problem under expected semantics. We will in the first place discuss the complexity of the problem and present an efficient, scalable and accurate algorithm to solve it. Beyond
solving the problem, we will compare this approach with a second approach, using probabilistic semantics, and show how the nature of mining results differs. Firstly, we illustrate the concepts defined above using an example:

![Fig. 2: example of isomorphisms and not isomorphisms for a certain graph](image)

Figure 2 illustrates the definition of isomorphism we introduced, using a certain graph $G$ and five subpatterns $S_1$, $S_2$, $S_3$, $S_4$, $S_5$. For $S_1$ and $S_2$, it is easy to see that they are subgraph isomorphic to $G$. They have respectively 4 and 2 embeddings in $G$. However $S_3$, $S_4$ and $S_5$ are not subgraph isomorphic to $G$: Considering $V$ the set of vertices of $G$ and $V'$ the set of vertices of each subpattern, there are sufficient reasons to confirm that there exists no injection $f : V' \rightarrow V$, respectively for $S_3$, $S_4$ and $S_5$. For example, there exists no edge in $G$ between to vertices labelled $A$ nor between two vertices labelled $B$ and $z$ does not exist in the set of labels of $G$.

Intuitively, testing the subgraph isomorphic property is equivalent to test if a structural pattern can be found in an exact graph or not. A major criticism we could express is that support counting does not take into account the multiple occurrences a particular subgraph can have in an implicated graph. Thus, there would be no difference in terms of mining results between two subgraphs that can be found in the same number of implicated graphs of an uncertain graph databases, even if one have more embedding in each implicated graph than the other one. This assumption may not be true in the real world and produce unrepresentative results. This may be a trail for further study, to redefine support counting.

![Fig. 3: example of expected support calculation for two subgraphs](image)
Now we would like to give an example of expected support calculation in Fig. 3. Let us use the two subgraph patterns $S_1$ and $S_2$ from Fig. 2, and an uncertain graph database $D$ containing only the uncertain graph $G_1$ from Fig. 1. Figure 3 represents all the implicated databases from $D$. Since $D$ has only one graph, it is equivalent to the 16 implicated graphs of $G_1$ labelled with their existence possibility as defined in (1). All the implicated graph, to which $S_1$ or $S_2$ are subgraph isomorphic, are highlighted respectively in red or blue.

Since there is only one uncertain graph in the database $D$, and according the support definition, the support of a subgraph in an implicated graph is equal to 1 if it is embedded in the implicated graph database, 0 otherwise. Thus, the expected support of a subgraph in $D$ is equal to the sum of the existence possibility of each implicated graph database in which it is embedded:

$$\text{esup}_D(S_1) = 0.018 + 0.048 + 0.042 + 0.112 + 0.042 + 0.072 + 0.112 + 0.168 + 0.168 = 0.782$$

and

$$\text{esup}_D(S_2) = 0.072 + 0.072 + 0.168 + 0.168 = 0.48.$$  

According to a user-specified support $\text{mins}_p$, subgraphs $S_1$ and $S_2$ will be output or not, e.g. for $\text{mins}_p = 0.4$ both of them will be output, for $\text{mins}_p = 0.7$ $S_1$ will be output and $S_2$ will not be output, for $\text{mins}_p = 0.8$ neither of them will be output.

2.3 Computational Complexity

Given an uncertain graph database $D$, a subgraph pattern $S$ in $D$ is said to occur in an uncertain graph $G$ belonging to $D$, denoted by $S \subseteq_U G$, if $S$ is subgraph isomorphic to at least one implicated graph of $G$. The probability of $S$ occurring in $G$ can be formulated as follows:

$$P(S \subseteq_U G) = \sum_{I \in \mathcal{I}(G)} P(G \Rightarrow I) \times \psi(I, S)$$  

where $\psi(I, S) = 1$ if $S$ is subgraph isomorphic to the implicated graph $I$ and $\psi(I, S) = 0$ if not. A major problem inherent from this model is that it is $\#P$-complete to compute the probability of a subgraph pattern occurring in an uncertain graph. It was proven in [1] by reducing the $\#P$-complete DNF counting problem to the problem of computing the probability, $P(S \subseteq_U G)$, of a subgraph pattern $S$ occurring in an uncertain graph $G$.

Also, the expected support can be expressed depending on the probability of occurring of a subgraph pattern in an uncertain graph database $D$. Indeed, it is the average of the probability of occurring in each implicated graph of $D$:

$$\text{esup}_D(S) = \frac{1}{|D|} \sum_{i=1}^{|D|} P(S \subseteq_U G)$$  

Thus, it is also $\#P$-complete to compute the expected support of a subgraph pattern in an uncertain graph database. Moreover, the number of frequent subgraph patterns in an uncertain graph database increases exponentially with the number of edges of the uncertain graph database, which implies that the complexity of the algorithm for mining subgraph patterns is exponential related to the size of the input. Formally, the problem of counting the number of frequent subgraph patterns is $\#P$-hard and as a result the problem of finding all frequent subgraph patterns is NP-hard. Please refer to [1] to obtain the formal proof of this statement and more generally to [12] to gain more information about complexity theory in the field of computer science.

2.4 MUSE: Mining Uncertain Subgraph Patterns

Because of the intractability of the frequent subgraph pattern mining problem, an approximate algorithm, which returns an approximate set of frequent subgraph patterns, was proposed. The complete algorithm
is called MUSE (Mining Uncertain Subgraph Patterns). Introducing a user-specified threshold \( \varepsilon \in [0, 1] \), so called relative error tolerance, MUSE shall output all subgraph patterns with expected support at least \( \minsup \), but all subgraph patterns with expected support less than \((1 - \varepsilon)\minsup\) shall not be output. Subgraphs, whose expected support is in \([(1 - \varepsilon)\minsup, \minsup]\), can be output or not.

The algorithm aims to fulfill 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.

We will show in the following subparts how these goals can be fulfilled and comment the performance evaluation provided in the paper that brought MUSE to public attention.

Method to complete Objective I :

To determine as efficiently as possible whether a subgraph pattern can be output or not, the expected support \( \esup_D(S) \) of a subgraph \( S \) in an uncertain graph database \( D \) can be approximated by a closed interval, denoted \([\esup_D(S), \esup_D(S)]\) such that the real expected support of \( S \) in \( D \) is included in this interval. The decision of outputting \( S \) can be taken by testing the following conditions:

**Condition 1:** If the upper border of the interval is greater than \( \minsup \) and the lower border greater than \((1 - \varepsilon)\minsup\), then \( S \) must be output, because it is certain that \( \esup_D(S) \) is greater than \((1 - \varepsilon)\minsup\) and it is likely that \( \esup_D(S) \) is greater than \( \minsup \).

**Condition 2:** If the upper border of the interval is lower than \( \minsup \), then \( S \) should not be output because it is certain that \( \esup_D(S) \) is lower than \( \minsup \).

**Condition 3:** If the upper border of the interval is greater than \( \minsup \) and the lower border smaller than \((1 - \varepsilon)\minsup\), then \( \esup_D(S) \) should be approximated by a smaller interval and the conditions should be tested again. Indeed, we are unable to determine if \( \esup_D(S) \) is likely to be greater than the support threshold.

Fig. 4: illustration of conditions for deciding whether to output a subgraph pattern or not

The three conditions and their associated decision are summarized in Fig. 4. Actually, we will never have to test condition 3 because the algorithm is designed to return an interval whose width is at most \( \varepsilon \times \minsup \). In that case, either condition 1 or condition 2 will be satisfied.

Method to complete Objective II :

For any subgraphs \( S \) in an uncertain graph database \( D \), any subgraphs \( S' \) of \( S \) are subgraph isomorphic to at least any implicated graphs of \( D \) to which \( S \) is subgraph isomorphic too. Therefore, it is easy to show that the expected support of a subgraph pattern \( S \) is greater or equal to the expected support of any
superpatterns of $S$ and smaller or equal to any subpatterns of $S$. This property is called the *apriori property* or *anti-monotonicity* of the expected support.

The apriori property of the expected support implies that all subpatterns of a frequent subgraph are also frequent, and that all superpatterns of an infrequent subgraph pattern are also infrequent. This property can be used to reduce the complexity of the mining algorithm by efficiently pruning sets of subgraphs from the ensemble of subgraphs that needs to be examined.

To take advantage of the apriori property of expected support, the ensemble of subgraphs of an uncertain graph database should be organized in a structure on which depth first search is systematically performed. All subgraph patterns can be organized as a directed acyclic graph, in which nodes represent subgraph patterns and edges represent *direct* subpattern relationship.

A subgraph pattern $S'$ is *direct* subpattern to a subgraph pattern $S$ if and only if $S'$ is subgraph isomorphic to $S$ and $|E(S')| = |E(S)| - 1$. A property from direct acyclic graph, which is problematic for our mining task, is that subgraph patterns may have more than one parent. This problem can be addressed by simplifying the directed acyclic graph to a tree, using some specific scheme. [1] proposed to use *DFS codes*, a state-of-the-art canonical graph coding scheme developed for frequent subgraph mining and exposed in [3]. Using DFS codes, the directed acyclic graph in Fig. 5 can be simplified by keeping only the solid arcs. Such structure is called a *search tree* of subgraph patterns.

![Diagram](image)

**Fig. 5:** the search tree of the subgraph pattern in $D$, containing an uncertain graph $G$, in Fig. 2

The major advantage from such a structure is that, by performing a depth-first search, each subgraph will be analyzed at most once, because it has only one parent. Moreover, if a subgraph pattern is known to be infrequent, then all its descendants do not need to be examined. This is true due to the apriori property of the expected support. This leads to efficient pruning in the search tree and can be used by MUSE to improve time efficiency. Here we show how MUSE takes advantage of this structure and the apriori property by giving a high level description of the algorithm:

**Step 1 : Initialisation**

Scan the edges in the uncertain graphs of $D$ to get all subgraph patterns consisting of only one edge, i.e. the smallest non trivial subgraph patterns, and push them into an empty stack $T$. 

Seminar: Similarity of Graphs

---

Edouard Fouché - IPD Böhm
Step 2: Compute approximate expected support

Pop the subgraph pattern $S$ from the top of $T$. Approximate the expected support, $\text{esup}_D(S)$, of $S$ in $D$ by an interval $[\text{esup}_D(S), \text{esup}_D(S)]$ such that the width of the interval is at most $\varepsilon \times \text{minsup}$.

Step 3: Output decision

Decide whether $S$ can be output or not by testing condition 1 and 2, as illustrated in Fig. 4. If $S$ cannot be output, then prune the subtree whose root is $S$ and go to step 4. If $S$ can be output, then output $S$ and generate all direct superpatterns of $S$. For each generated superpattern $S'$, if $S'$ is a child of $S$ in the search tree, then push it into $T$.

Step 4: Stopping condition

If $T$ is empty, then terminate, otherwise go to step 2.

The complexity of the mining algorithm was reduced by fulfilling objectives 1 and 2. However, the challenge that remains is how to compute the approximate expected support. Addressing this problem is substantial for reducing the complexity of the mining algorithm and will be discussed in the following part.

2.5 Algorithm for Computing Expected Support

According to (5), the expected support of a subgraph $S$ corresponds to the average of the probability, denoted $P(S \subseteq U G)$, of $S$ occurring in every uncertain graph $G \in D$. However computing $P(S \subseteq U G)$ is #P-complete. To overcome this difficulty, an exact algorithm to exactly compute $P(S \subseteq U G)$ for small problem instances and an approximation algorithm to approximate $P(S \subseteq U G)$ for large instances is proposed.

Based on its definition, i.e. Eq 4, computing $P(S \subseteq U G)$ implies computing the probability distribution $P(G \Rightarrow I)$ over all implicated graphs of $G$ and test if $S$ is subgraph isomorphic to each of them. Since there are $2^{|E(G)|}$ implicated graphs in $G$, it is intractable. In practice, this cannot scale if $G$ has more than 30 edges.

In the exact algorithm, the problem of computing $P(S \subseteq U G)$ is transformed to the DNF counting problem and can be solved in $\Theta(2^{n-1} n |E(S)|)$ where $n$ is the number of embedding of a subgraph $S$ in an uncertain graph $G$. To get more information about the fundamental technique of DNF counting used in this algorithm, please refer to [1].

In the approximation algorithm, the same method is used, i.e. the problem is transformed to the DNF counting problem by constructing a DNF Formula $F$, but instead of exactly computing the satisfaction probability $P(F)$ in exponential time, it is approximated in polynomial time by an interval $[l, u]$ such that the width of $[l, u]$ is at most $\varepsilon \times \text{minsup}$. This computation is achieved using the fully polynomial randomized approximation scheme (FPRAS) proposed by Karp and Luby [13]. For a given DNF formula $F$, the FPRAS can approximate in polynomial time $P(F)$ by an interval $[l, u]$ with $|u - l| \leq \varepsilon$ with probability $1 - \delta$, where $\varepsilon$ an absolute error and $\delta$ a real number in the interval $[0, 1]$, both of them specified as a parameter by the user. The expected time complexity of the approximation algorithm is $O(N n|E(S)|)$ where $N$ is the number of loops carried out by the FPRAS. For more information about the approximation algorithm please refer to [1] and to [13] for more information about the FPRAS.

MUSE uses both of these algorithms and is able to determine which of them should be used depending on the subgraph that is being analysed. Indeed, due to its exponential runtime the exact algorithm is faster for small problem instances but drastically slower for bigger instances. The complete algorithm of $MUSE$ is described in [1].
2.6 Performance Evaluation

Some experimentations were performed and presented in [1] to show the efficiency, quality and scalability of MUSE, as well as the impact of uncertainties on the efficiency. MUSE was implemented in C, and all experiments were performed using a real uncertain graph database, obtained from the STRING database, which contains the PPI networks of six organisms. The experiments were performed on an IBM ThinkPad T61 notebook with 2GHz CPU and 2GB RAM, running Windows XP.

![Performance Evaluation](image1)

**Fig. 6:** performance evaluation of time efficiency, related to $\text{minsup}$, $\epsilon$ and $\delta$

Figure 6 represents some experimental results monitoring the execution time of MUSE related to its parameters. Figure 6(a) shows that the execution time decreases drastically when the support threshold, $\text{minsup}$, increases. This is due to the fact that the number of subgraphs, that needs to be examined and output, decreases when $\text{minsup}$ becomes larger. Figure 6(b) shows that the execution time decreases rapidly as $\epsilon$ becomes larger. The reason is that the execution time of the approximation algorithm is proportional to the inverse of $\epsilon^2$. Some more experimental results were provided by the author that shows a good accuracy: 92% to 97% of frequent subgraph are effectively output, while the fraction of infrequent output subgraphs only depends on the parameter $\epsilon$, the relative error tolerance. The scalability of MUSE was also proven to be linear in execution time and memory usage with respect to the number of graphs in the database. The experimental results can be seen in Fig. 7, they show that MUSE is very scalable to large uncertain graph databases.

![Scalability](image2)

**Fig. 7:** scalability of MUSE related to the number of uncertain graphs.
3 FREQUENT SUBGRAPH MINING UNDER PROBABILISTIC SEMANTICS

The frequent subgraph mining problem was also studied by Zou et al. under probabilistic semantics. This approach shares a number of definitions and concepts with the first one, thus it can be seen as an extension of the first model. In this part, we are going to present the probabilistic semantic approach and highlight how it differs from the expected semantics.

3.1 Extension of Problem Statement and Model

The probabilistic semantic uses the same model than the previously described approach, except that probability on vertices are introduced and taken into account. As a consequence, an uncertain graph is a system $G = (V, E, \Sigma, L_V, L_E, P_V, P_E)$ where $L_V : V \cup E \rightarrow \Sigma$ is a function assigning labels to vertices and $L_E : V \cup E \rightarrow \Sigma$ a function assigning labels to edges, $P_V : E \rightarrow (0, 1]$ is a function assigning existence possibility values to vertices and $P_E : E \rightarrow (0, 1]$ a function assigning conditional existence probability values to edges given their endpoints. Note that if the existence possibilities of all vertices is set to 1, then this uncertain graph model is equivalent to model described under expected semantics. Thus, it has more expressivity by allowing vertices to exist or not.

However, the extension of the model may induce a higher computational complexity since the number of implicated graphs for an uncertain graph is duplicated. Let $\text{Imp}(G)$ be the set of implicated graphs of an uncertain graph $G$. In the worst case, i.e. $G$ is a fully connected graph, the cardinality of $\text{Imp}(G)$ is equal to $|\text{Imp}(G)| = \sum_{i=1}^{V} \binom{|V|}{i} 2^{(|V|-i)/2}$

This approach is also based on the assumption that the existence probabilities of vertices and the conditional existence probabilities of edges of an uncertain graph are mutually independent. Which oversimplify the problem, but may not be satisfied in reality. A probability distribution over $\text{Imp}(G)$ can be computed using the following formula:

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

Finally, based on the assumption that graphs in an uncertain graph database are mutually independent, we can derive the following formula :

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

Note that the last term in (7) refers to the trivial graphs, i.e. graphs that don’t have any edges nor vertices, that we should take into account when computing the probability distribution.

A new measure called $\varphi$ – frequent probability, denoted $P_{D,\varphi}(S)$, can be introduced, based on the probability distribution of implicated graph database and the traditional support definition are expressed in section 2.2:

$$P_{D,\varphi}(S) = \sum_{D' \in \text{Imp}(D), \sup_{D'}(S) \geq \varphi} P(D \Rightarrow D')$$
The $\phi$-frequent probability of a pattern $S$ in an uncertain graph database $D$ corresponds to the sum of the existence possibility values of all implicated graph databases $D'$ in which the support of $S$ is greater than $\phi$. We say that a subgraph is $(\phi, \tau)$-probabilistic frequent when the $\phi$-frequent probability of $S$ is greater than a user-specified confidence threshold $0 < \tau \leq 1$. Thus, a main difference with the first approach is that the user has to set two different parameters $\phi$ and $\tau$, which can be respectively regarded as support and confidence of discovered graph structures.

3.2 Mining Frequent Subgraphs under Probabilistic Semantics

Since the approach for finding frequent subgraph in probabilistic semantic is highly similar to the expected semantic approach, we will not go into much detail, but rather provide a high-level description of the algorithm, for further details, please refer to [2].

The algorithm needs as input an uncertain graph database $D$, a support threshold $\phi$, a confidence threshold $\tau$ and an error tolerance $\epsilon$. It performs then the following steps:

**Step 1 : Organize all subgraphs in $D$ into a search tree**

All possible subgraphs are organized into a search tree, in which each node represents a subgraph and each node is a direct subgraph isomorphic to all its children, i.e. it is subgraph isomorphic and it has only one less edge than all of them. Similarly to MUSE, the search tree is simplified using DFS codes, so that each node has only one parent. This ensures effective pruning of the search space in the next step.

**Step 2 : Examine the tree in depth-first order**

The algorithm systematically performs a depth-first-search in the tree and determines in polynomial time for each subgraph $S$ whether it has a $\phi$-frequent probability at least $\tau - \epsilon$ and probably at least $\tau$. If the answer is “yes”, then $S$ shall be output and its descendants will be examined in depth-first order. If the answer is “no”, then $S$ shall not be output, and all descendants of $S$ are infrequent. As a consequence, the subtree rooted in $S$ can be pruned from the search space. This is possible because $\phi$-frequent probability respects the apriori property.

3.3 An Approximation Algorithm for Computing the $\phi$-frequent probability

For each subgraph $S$ examined in step 2, the algorithm approximates the $\phi$-frequent probability, denoted $P_{D,\phi}(S)$, using a dynamic programming-based method that produces an interval $[p_l, p_u]$ such that $|p_u - p_l| \leq \epsilon$ and that $P_{D,\phi}(S) \in [p_l, p_u]$. The asymptotical execution time of this step dominates the rest of the algorithm and corresponds to an asymptotical time complexity of $O(m^2 n^3 \log(n))$ where $n$ corresponds to the number of vertices and $m$ the number of edges in the uncertain graph database $D$. For more information about this method, please refer to [2].

3.4 Performance Evaluation

The performance evaluation of this approach was performed on the same uncertain graph database as the first one. On the one hand, we are able to compare - with some reservations - the performances of both approaches. On the other hand, there are no probabilities on the vertices in this database. As a consequence, the extended model is reduced to the first simple model, by simply assigning an existence possibility of 100% to each vertex. We may even be surprised that no additional tests were performed with the introduction of uncertainties on vertices. Thus, it is impossible to know if considering uncertainties on vertices is interesting and realistic in terms of execution time.

In Fig. 8 some experimental results represent the execution time related to the two user-specified thresholds $\phi$ and $\tau$. We labelled in black for each level the number of subgraphs visited by the algorithm and in
Fig. 8: impact of threshold $\phi$ and $\tau$ on execution time

The performance evaluation shows that execution time drops drastically when threshold $\varepsilon$ and $\tau$ increase. The reason is that fewer subgraphs need to be examined and output, as we can see on the labels.

However, there was no experiment to show the scalability of this approach. This may arouse suspicion, because the authors performed a scalability experiment on the first approach. Scalability may be a weakness of analysing under probabilistic semantics.

4 COMPARISON OF EXPECTED AND PROBABILISTIC SEMANTICS

Fig. 9: comparison of the efficiency of the two approach

Combining the experimental results provided by both studies, we tried to compare the performance of both of them. We define a measure that we call efficiency. It corresponds to the number of graphs output divided by the execution time. Intuitively, “for each second of computation, how many subgraphs are output”. Then we display the results on the same plot for comparison (Figure 9). The data that produces this plot was deduced from the plots we may find in paper [1] and [2]. No original values were provided, so they are
prone to imprecision and measurement errors. The blue curve corresponds to the results for MUSE, the red curve to the results obtained with the probabilistic approach.

The comparative study shows similar results for a varying error tolerance, but in Fig. 9(b), we can see that the performance of MUSE becomes dramatically better when the support threshold increases. We have to be careful with these results, because it is not even sure that we can compare the parameters minsup and \( \varphi \), since the mining result in the second approach also depends on the value of the confidence threshold \( \tau \), which is fixed to 0.9 in all experiments. However, the experimentation for the second approach was performed on a different machine, with better expected performance (2.4GHz CPU against 2GHz CPU, 8GB RAM against 2GB RAM). By chance, this only comforts the hypothesis we may deduce from these results, i.e. MUSE is more efficient.

5 CONCLUSION

Mining frequent subgraphs in uncertain graphs has been addressed recently. Even if the problem of mining frequent subgraphs in certain graphs has been extensively studied, a lot of definitions and concepts from previous studies do not hold because of the introduction of uncertainties in graphs. So far, two different approaches attempt to solve the problem from two different angles: expected semantics and probabilistic semantics. Both of them define a new measure for quantifying the frequency of subgraphs in uncertain graph databases, respectively called expected support and \( \varphi \) – frequent probability.

Since the problem of mining frequent subgraphs in uncertain graphs is NP-hard, only an approximate set of frequent subgraphs, using an approximation algorithm, can be obtained in a reasonable period of time. Both approaches support the mining process by using depth-first-search strategy on a search tree structure and some state-of-the-art techniques such as DFS codes [3], to enable an efficient discovery of frequent subgraph patterns.

However, the two approaches differ in terms of mining result and execution time. Expected semantics approach seems to be better suited to use as an exploratory tool, because it finds items whose multiplicity in a randomly generated world is expected to reach a certain quantity. As a consequence, it is more likely to return common structural patterns among a set of graphs. This may be useful as a user to get rapidly an insight on the nature of the data. Probabilistic semantics approach seems to be designed for the extraction of features, since it captures items that are likely to be frequent in a randomly generated world. Due to the combination of two parameters, corresponding to “support” and “confidence” of mining results, it seems more flexible. For example, a user may be interested in finding patterns that are very likely to be embedded in a small portion of the database (high confidence but low support) or as a contrary, finding patterns that are embedded in a large portion of the database, but unlikely to exist (low confidence but high support). Nevertheless, such an exploration method may require more knowledge from the user.

This would be helpful to develop a framework to support user discovery, that interactively runs both algorithms and automatically sets parameters, so that they are likely to return interesting patterns. The first approach may also be used as a preprocessing step for the second one, since it returns more “general” results and seems computationally less expensive, even though it cannot handle uncertainties on vertices.

A few criticisms can be expressed about both approaches. First of all they assume mutual independence between edges, vertices and graphs in uncertain graph databases. It is not sure that this assumption is reasonable in real world applications except for some particular cases. Moreover, both frequency measures do not take into account multiple embedding of subgraphs in particular uncertain graphs. Furthermore, we regret that the authors did not experiment the scalability of the second approach. Finally, no benefits have been shown from the extension of the uncertain graph model, since no uncertain graph databases with uncertainties on vertices have been used for testing. If the reason is that such an extension has no real world application, then the complexity of the probabilistic semantics approach could be reduced. Otherwise, the model for the expect semantics approach could be extended in the same way.
ACKNOWLEDGMENTS

We thank Professor Klemens Böhm and Hoang Vu Nguyen, from the Institute for Program Structures and Data Organisation at KIT, for providing helpful feedback regarding this paper and the corresponding presentation, which was given in the context of this seminar. We are also grateful to Karl Lorey and Charlotte Royer for the insightful comments and advice. For any comments or suggestions, please feel free to contact the author using the following address: contact@edouardfouche.com

References

1. Z. Zou, J. Li, H. Gao, and S. Zhang. Frequent Subgraph Pattern Mining on Uncertain Graph Data. In proceeding of the CIKM’09.
3. X. Yan and J. Han. gSpan: Graph-based Substructure Pattern Mining. In proceeding of the ICDM’02.
5. Z. Zou, J. Li, H. Gao, and S. Zhang. Mining Frequent Subgraph Patterns from Uncertain Graph Data. In the proceeding of TKDE’10.