Discovering Highly Reliable Subgraphs in Uncertain Graphs *

Ruoming Jin
Kent State University
Kent, OH, USA
jin@cs.kent.edu

Lin Liu
Kent State University
Kent, OH, USA
lliu@cs.kent.edu

Charu C. Aggarwal
IBM T. J. Watson Research Ctr
Hawthorne, NY, USA
charu@us.ibm.com

ABSTRACT

In this paper, we investigate the *highly reliable subgraph* problem, which arises in the context of uncertain graphs. This problem attempts to identify all induced subgraphs for which the probability of connectivity being maintained under uncertainty is higher than a given threshold. This problem arises in a wide range of network applications, such as protein-complex discovery, network routing, and social network analysis. Since exact discovery may be computationally intractable, we introduce a novel sampling scheme which enables approximate discovery of highly reliable subgraphs with high probability. Furthermore, we transform the core mining task into a new *frequent cohesive set problem* in deterministic graphs. Such transformation enables the development of an efficient two-stage approach which combines novel *peeling techniques* for maximal set discovery with depth-first search for further enumeration. We demonstrate the effectiveness and efficiency of the proposed algorithms on real and synthetic data sets.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms, Theory

Keywords

Uncertain graph, Reliable subgraph, Frequent cohesive set

1. INTRODUCTION

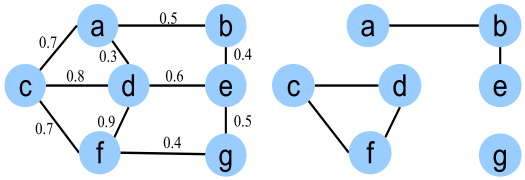
Networks have evolved into a unified conceptual abstract to model both natural and man-made complex systems such as the web, social networks, and cellular systems. Such networks often have inherent uncertainty. In a telecommunication or electrical network, a link can be unreliable and may fail with certain probability [11, 31]. In a protein interaction network, the pairwise interaction is derived from (approximate) statistical models [4, 3, 21]; in a social network, trust and influence issues may impact the likelihood of social interactions [12, 26, 1]. The *uncertain graph model* is a convenient framework to address such scenarios [30, 38, 41, 40, 39] in a unified way. In this model, each edge is associated with an *edge existence probability* to quantify the likelihood that this edge exists in the graph. An uncertain graph is also referred to as a *probabilistic graph* [2, 17].

It is evident that the connectivity of the network is a complex probabilistic function of the network topology and edge uncertainty. Such problems are notoriously difficult [34, 5] because of the combinatorially large number of possible instantiations of an uncertain graph. For example, even the simple problem of pairwise vertex reachability, which can be easily computed in linear time in deterministic graphs, becomes a $\#P$-complete problem in the uncertain scenario [5]. The *network reliability problem* [10] is a generalization of pairwise reachability, in which the goal is to determine the probability that *all pairs of nodes* are reachable from one another. In other words, network reliability measures the probability that the entire graph remains connected under uncertainty. The basic reachability problem is also referred to as *two-terminal network reliability* and the latter problem is referred to as *all-terminal network reliability*, which is also $\#P$-complete [5]. As we will see, the reliable subgraph problem addressed in this paper is an even further generalization of these problems, as it relates to subgraph **discovery** satisfying the reliability property. Network reliability algorithms have numerous applications in communication networks [11, 31], biological networks [3], and social networks [33, 35].

This paper uses a pattern mining approach towards reliable subgraph discovery, which is independent of specific sets of vertices. We address the problem of determining all *highly reliable sets of vertices* for which the induced subgraphs have network reliability above a user-specified threshold. The problem of discovering all highly reliable subgraphs can be found in a wide range of network applications. In a protein-protein interaction network, the highly reliable subgraphs can be used for identifying the core of protein complexes [3]. In a telecommunication network, the highly reliable subgraphs can serve as the basic constructs to help connect a source and a destination [29]. In social networks with edge uncertainty, highly reliable subgraphs can help discover cohesive groups [35]. The highly reliable subgraph problem on uncertain networks can be considered analogous to dense component mining in the deterministic scenario [28]. However, the combination of network topology and edge probabilities make this problem much more challenging, because a structurally dense component with low probability edges may not be reliable, and vice versa. In general, the subtle effects of different levels of uncertainty in different structural portions of the network need to be addressed by any algorithm for reliable sub-

*R. Jin and L. Liu were partially supported by the National Science Foundation, under CAREER grant IIS-0953950.

KDD'11, August 21–24, 2011, San Diego, California, USA.



(a) Uncertain Graph $\mathcal{G}$ (b) G_1 with 0.00254016

Figure 1: Running Example

graph mining. In order to address these goals, we make the following contributions in this paper:

1. We introduce and formally define the problem of discovering all highly reliable subgraphs in uncertain graphs (Section 2).
2. We propose a sampling scheme, which enables approximate discovery of highly reliable subgraphs with guaranteed probabilistic accuracy. We transform the uncertain graph mining problem into a new *frequent cohesive set discovery* problem in deterministic graphs (Section 3).
3. We present a two-stage approach to discover all frequent cohesive sets. The algorithm combines an efficient top-down *peeling* approach in the first stage (to discover all *maximal* frequent cohesive sets) with a fast depth-first-search (DFS) mining procedure to discover the remaining non-maximal frequent cohesive sets (Section 4).
4. We demonstrate the effectiveness and efficiency of the proposed algorithms on real and synthetic data sets (Section 5).

2. PROBLEM FORMULATION

We denote an *uncertain directed graph* by $\mathcal{G} = (V, E, P)$, where V is a set of vertices, E is a set of edges, and $P : E \rightarrow (0, 1]$ is a function that determines the probability of existence of edge e in the graph. It is assumed that the uncertainty variables of different edges are independent of one another, though our approach also applies in principle to the *conditional probability model* [30], as long as a computational method for generating appropriate samples of the uncertain graph is available.

A *possible graph* $G = (V_G, E_G)$ of an uncertain graph $\mathcal{G}$ is a deterministic graph which is a possible *outcome* of the random variables representing the edges of the probabilistic graph $\mathcal{G}$. We denote the possible graph by $G \sqsubseteq \mathcal{G}$. The graph G contains a subset of edges of $\mathcal{G}$. In other words, we have $E_G \subseteq \mathcal{G}$. The total number of such possible graphs is $2^{|E|}$, because for each edge, we have two cases as to whether or not that edge is present in the graph. The probability of sampling the graph G from the random variables representing the uncertain graph $\mathcal{G}$ is given by the following sampling or realization probability $\mathbf{Pr}[G]$:

$$\mathbf{Pr}[G] = \prod_{e \in E_G} p(e) \prod_{e \in E \setminus E_G} (1 - p(e)).$$

We present an example of an uncertain graph $\mathcal{G}$ and its possible realization G_1 in Figure 1. The uncertain graph $\mathcal{G}$ has 2^{10} possible outcomes, and the sampling probability of G_1 is defined as follows:

$$\mathbf{Pr}[G_1] = p(a,b)p(b,e)p(c,d)p(c,f)p(d,f)(1-p(a,c)) \times (1-p(a,d))(1-p(d,e))(1-p(e,g))(1-p(f,g)) = 0.00254016$$

The network reliability [10] measures the likelihood that the entire graph is connected and can be formally defined as follows:

DEFINITION 1. *(**Network Reliability**) Given an uncertain graph $\mathcal{G} = (V, E, P)$, its network reliability $\mathbf{R}[\mathcal{G}]$ is defined as the probability that its sampled realizations remain connected. In other words, we have:*

$$\mathbf{R}[\mathcal{G}] = \sum_{G \sqsubseteq \mathcal{G}} \mathbf{I}(G) \cdot \mathbf{Pr}[G], \text{ where } \mathbf{I}(G) \text{ is an indicator function,}$$

$$\mathbf{I}(G) = \begin{cases} 1, & \text{if } G \text{ is connected ;} \\ 0, & \text{otherwise} \end{cases}$$

The definition can be easily generalized to the induced subgraph $\mathcal{G}[V_s]$ for a subset of vertices $V_s \subseteq V$:

$$\mathbf{R}[\mathcal{G}[V_s]] = \sum_{G \sqsubseteq \mathcal{G}} \mathbf{I}(G[V_s]) \cdot \mathbf{Pr}[G], \text{ where } G[V_s] \text{ is the induced}$$

subgraph of V_s. For simplicity, we also refer to $\mathbf{R}[\mathcal{G}[V_s]]$ as the *subgraph reliability* with respect to V_s.

Next, we introduce the problem of discovering all highly reliable subgraphs in an uncertain graph:

DEFINITION 2. *(**Highly Reliable Subgraph (HRS) Problem**) Given an uncertain graph $\mathcal{G} = (V, E, P)$ and a reliability threshold $\alpha \in [0, 1]$, determine all induced subgraphs whose network reliability is at least α.*

In other words, we have $\mathbf{R}[\mathcal{G}[V_s]] \geq \alpha$. The resulting set is denoted by $\mathcal{S}_\alpha$. We make the important observation that the *downward closure property* does *not* hold for reliable subgraphs:

LEMMA 1. *Given an uncertain graph $\mathcal{G}$ and two subsets V_s and V_s' where $V_s \supset V_s'$, we cannot claim either $\mathbf{R}[\mathcal{G}[V_s]] \geq \mathbf{R}[\mathcal{G}[V_s']]$ or $\mathbf{R}[\mathcal{G}[V_s]] \leq \mathbf{R}[\mathcal{G}[V_s']]$.*

For instance, in the uncertain graph $\mathcal{G}$ depicted in Figure 1, $\mathbf{R}[\mathcal{G}[\{c,d,f\}]] = 0.902 > \mathbf{R}[\mathcal{G}[\{d,f\}]] = 0.9$ and we have $\mathbf{R}[\mathcal{G}[\{b,e,g\}]] = 0.2 < \mathbf{R}[\mathcal{G}[\{e,g\}]] = 0.5$. The lack of downward closure creates a challenge from an algorithmic perspective.

In addition, we can use a simple property of the uncertain graph in order to preprocess it and drop some of the edges in order to reduce computational complexity. This property is as follows: *Given an uncertain graph $\mathcal{G} = (V, E, P)$ and a reliability threshold α, for an induced subgraph $\mathcal{G}[V_s]$ of $\mathcal{G}$, if there is an edge cut $C \subseteq E(\mathcal{G}[V_s])$ in $\mathcal{G}[V_s]$ (removing all edges in C makes $\mathcal{G}[V_s]$ disconnected), such that $p(C) = \sum_{e \in C} p(e) < \alpha$, then $\mathbf{R}[\mathcal{G}[V_s]] < \alpha$.* Given this, we can further observe the following:

LEMMA 2. *(**Edge-Cut Lemma**) Given an uncertain graph $\mathcal{G} = (V, E, P)$ and a reliability threshold α, for any edge cut C in $\mathcal{G}$, $C \subseteq E$ which breaks the uncertain graph into two parts $\mathcal{G}[V_1]$ and $\mathcal{G}[V_2]$ ($V_1 \cup V_2 = V$ and $C \subseteq V_1 \times V_2$), then if $p(C) = \sum_{e \in C} p(e)$, then, for any high reliable subgraph (HRS) $\mathcal{G}[V_s]$, either we have $V_s \subseteq V_1$ or $V_s \subseteq V_2$. In other words, there is no HRS $\mathcal{G}[V_s]$ with $V_s \cap V_1 \neq \emptyset$ and $V_s \cap V_2 \neq \emptyset$.*

The implication of this lemma is that if for any cut C in uncertain graph $\mathcal{G}$ with $p(C) < \alpha$, then, we can safely drop all the edges in C without missing any HRS. Clearly, this reduces the number of candidate sets. However, finding all such cuts with value at most α is computationally prohibitive. In this work, we apply a simple linear min-cut algorithm [32] starting from each vertex in the uncertain graph in order to drop edges. This algorithm starts from an arbitrary vertex (a single vertex set) in a graph, and then iteratively selects a vertex most tightly connected to the current set of vertices and add it into the set until we find a cut C with $p(C)$ less than α. At this point, we drop all the edges in the cut. We utilize this procedure to preprocess the uncertain graph $\mathcal{G}$ by repeating the procedure for all vertices.

We note that all singleton vertex sets are always highly reliable, and a pair of vertices is a highly reliable set if and only if they are connected by an edge with probability at least α. Therefore, we will focus on the interesting case of finding all highly reliable vertex sets in $\mathcal{S}_\alpha$ with set size no less than 3. Finally, since each induced subgraph is uniquely determined by its vertex set, we report only vertex sets in $\mathcal{S}_\alpha$, and interchangeably use the terms "vertex sets" and "induced subgraphs" in this paper.

3. SAMPLING APPROXIMATION SCHEME

Since the network reliability problem is #P-complete, the generalized problem studied in this paper is at least as intractable. The methods available for exact determination of subgraph reliability are designed for cases where graphs are very small (tens of vertices at most) [31]. We note that the problem of subgraph reliability estimation and subgraph pattern discovery have different levels of difficulty in terms of solvability. The former problem can of course be partially addressed by allowing approximate estimation through Monte-Carlo sampling; however we will see that it continues to be a challenge to avoid uncontrolled false positives or negatives in the pattern discovery version of the problem. The sampling approach for subgraph reliability estimation is as follows:
(1) We first sample N possible graphs, $G_1, G_2, \cdots, G_N$ of $\mathcal{G}$ according to edge probability function P; and (2) for any subset of vertices V_s, we compute the indicator function $\mathbf{I}(G_i[V_s])$ (whether the induced subgraph $G_i[V_s]$ itself is a connected component) for each sample graph G_i. Given this, the sampling estimator ($\widehat{\mathbf{R}}[\mathcal{G}[V_s]]$) of the subgraph reliability is as follows:

$$\mathbf{R}[\mathcal{G}[V_s]] \approx \widehat{\mathbf{R}}[\mathcal{G}[V_s]] = \frac{\sum_{i=1}^{N} \mathbf{I}(G_i[V_s])}{N}$$

The sampling estimator $\widehat{\mathbf{R}}[\mathcal{G}[V_s]]$ is an *unbiased* estimator of the subgraph reliability, i.e., $E(\widehat{\mathbf{R}}[\mathcal{G}[V_s]]) = \mathbf{R}[\mathcal{G}[V_s]]$. More importantly, by applying the Chernoff-Hoeffding Bound [9, 19], we can determine whether an induced subgraph is highly reliable with high probability:

LEMMA 3. *With sample size $N \geq \frac{2}{\epsilon^2} \ln \frac{2}{\delta}$, for any subset of vertices V_s, $Pr(|\widehat{\mathbf{R}}[\mathcal{G}[V_s]] - \mathbf{R}[\mathcal{G}[V_s]]| \geq \epsilon) \leq \delta$.*

We would naturally like to use this approach for determining the highly reliable set $\mathcal{S}_\alpha$. However, from a set mining perspective, it is hard to probabilistically control the number of false positives or negatives. This is because the use of sampling to determine the reliability of an induced subgraph is a multiple hypothesis test, and it is inherently difficult to provide guarantees in such cases [6, 7]. Some recent results have shown how to effectively use multiple hypothesis tests for the special case of the frequent pattern mining problem [15, 27], but these methods cannot be used to provide guarantees for our problem.

To deal with this challenge, we utilize two sets to approximate $\mathcal{S}_\alpha$: *(a) the first set $\overline{S}$ which tries to maximize the recall of discovering highly reliable subgraphs; (b) the second set $\underline{S}$ which tries to maximize the precision of discovery.* These two sets provide flexible and complementary choices for different applications. In cases where we do not wish to miss any highly reliable subgraphs (false negatives), the first set can be used, and then false positives can be filtered out. If the application is designed for discovering some of the very highly reliable subgraphs, the second set is more helpful. When the two sets are similar, it is evident that it is possible to achieve both high precision and recall. We will design an approach which achieves this goal.

Sampling-based Approach for Highly Reliable Set Discovery: Our sampling-based approach utilizes additional parameters ϵ and δ for controlling the discovery results. The approach consists of two steps:

Step 1: *Sample $N_1 = \frac{2}{\epsilon^2} \ln \frac{2}{\delta}$ possible graphs of $\mathcal{G}$, denoted as dataset D_1, discovering all induced subgraphs $\mathcal{G}[V_s]$ with reliability $\widehat{\mathbf{R}}[\mathcal{G}[V_s]] \geq \alpha - \epsilon$ using N_1 sampled graphs:*

$$\overline{S} = \{\mathcal{G}[V_s] | \widehat{\mathbf{R}}[\mathcal{G}[V_s]] \geq \alpha - \epsilon\}$$

Step 2: *Sample $N_2 = \frac{2}{\epsilon^2} \ln \frac{2|\overline{S}|}{\delta}$ possible graphs of $\mathcal{G}$, denoted as dataset D_2, discovering all induced subgraphs $\mathcal{G}[V_s]$ with reliability $\widehat{\mathbf{R}}[\mathcal{G}[V_s]] \geq \alpha + \epsilon$ using N_2 sampled graphs and $\mathcal{G}[V_s] \in \overline{S}$:*

$$\underline{S} = \{\mathcal{G}[V_s] | \widehat{\mathbf{R}}[\mathcal{G}[V_s]] \geq \alpha + \epsilon\} \cap \overline{S}$$

We make the following observations about this approach.

THEOREM 1. ***(Precision and Recall)** The sets $\overline{S}$ and $\underline{S}$ produced by the sampling procedure have the following properties:*

1. *The expected fraction of missed highly reliable subgraphs is at most δ. In other words, we have $E(\frac{|\mathcal{S}_\alpha \setminus \overline{S}|}{|\mathcal{S}_\alpha|}) \leq \delta$;*
2. *With probability at least $1 - \delta$, all induced subgraph in $\underline{S}$ are highly reliable subgraphs: $Pr(\forall \mathcal{G}[V_s] \in \underline{S}, \mathbf{R}[\mathcal{G}[V_s]] \geq \alpha) \geq 1 - \delta$ and $Pr(|\underline{S} \cap \mathcal{S}_\alpha| / |\underline{S}| = 1) \geq 1 - \delta$;*
3. *With probability at least $1 - \delta$, the precision of $\overline{S}$ is no less than $\beta = \frac{|S|}{|\overline{S}|}$ ($\underline{S} \subseteq \overline{S}$): $Pr(|\mathcal{S}_\alpha \cap \overline{S}| / |\overline{S}| \geq \beta) \geq 1 - \delta$. When β is close to 1, the two sets become similar and the precision increases;*
4. *The expected fraction of missed highly reliable subgraphs in $\overline{S}$ is no higher than $\delta + 1 - \beta + \delta\beta/|\overline{S}| \approx \delta + 1 - \beta$: $E(\frac{|\mathcal{S}_\alpha \setminus \underline{S}|}{|\mathcal{S}_\alpha|}) \leq \delta + 1 - \beta + \delta\beta/|\overline{S}| \approx \delta + 1 - \beta$. When β is close to 1, the expected false negative rate is small.*

We omit the detailed proofs since these are direct applications of the Chernoff-Hoeffding bounds. It is evident from Theorem 1 that δ is directly responsible for controlling the precision and recall of the reliable subgraph discovery algorithm. The parameter ϵ needs further explanation. Though it does not directly appear in the precision and recall formulas, it helps to control the difference between $\overline{S}$ and $\underline{S}$. In other words, when ϵ decreases, β tends to increase. However, since the required sample sizes N_1 and N_2 are inversely related to the square of ϵ, it may be important to balance between computational cost and approximation quality. We will examine these tradeoffs in detail in the experimental section.

An important observation is that the precision and recall are controlled in this approach by different probabilistic criteria. Specifically, in order to control recall, the expected false negative rate is applied instead of the standard Bonferroni correction for multiple comparison [15]. This is because of the difficulty in determining the size of the result set $\mathcal{S}_\alpha$. To deal with this problem, we simply employ the expected false negative rate in the first step. Once we have a candidate set $\overline{S}$, we are able to utilize a more strict Bonferroni correction approach [15] in the second step for controlling precision. It is important to remember that the approach (and the corresponding theoretical results) can be applied to an uncertain graph in which edge probabilities are not independent, as long as appropriate samples of the graph can be generated. For this purpose, an independent set of possible graphs can be generated from the uncertain graph by a Gibbs sampler or a Markov Chain Monte Carlo technique [30].

Discovering Frequent Cohesive Sets: The aforementioned sampling approximation scheme results in a new graph mining problem, which we call the *Frequent Cohesive Set* (FCS) problem:

DEFINITION 3. *(**Frequent Cohesive Set Problem**) Given a graph G and a subset of vertices $V_s \subseteq V[G]$, if its induced subgraph $G[V_s]$ forms one connected component, then V_s is referred to as a* **cohesive set** *in G. Given a set of graphs $D = \{G_1, G_2, \cdots, G_N\}$ with vertices $V(G_1) = V(G_2) = \cdots = V(G_N) = V$ and a minimal support threshold θ, a frequent cohesive set (FSC) is any subset of vertices $V_s \subseteq V$ that is a cohesive set in at least $\theta \cdot N$ graphs.*

In order to be consistent with the problem of highly reliable subgraph discovery, we denote the frequency of V_s being cohesive as $\widehat{\mathbf{R}}[V_s]$. In addition, the **maximal frequent cohesive set (MFCS)** problem identifies all frequent cohesive sets in D, for which none of their supersets are cohesive. In the aforementioned sampling approach, the first step is the key, and it is needed to solve the MFCS problem. Specifically, the dataset D_1 consists of N_1 graphs (Step 1 in the sampling approach), and the generation of the approximate set $\overline{S}$ corresponds to the discovery of all frequent cohesive sets in D_1 with minimum support $\theta = \alpha - \epsilon$. Once we have determined $\overline{S}$, we only need to determine whether each of the vertex sets is frequently cohesive in the new sampling dataset D_2 (Step 2). Since connectivity can be checked for each induced subgraph in linear time [20], this can be done rather quickly. Due to the simplicity of the second step, we omit further discussion and focus on the first step of discovering all frequent cohesive sets in a graph database.

Interestingly, to the best of our knowledge, this problem has not been studied before. The closest is Yan *et al.*'s work [36] on mining closed frequent subgraphs with connectivity constraints. The main difference is that in our problem, there is no isomorphism requirement on the induced subgraphs, a fact which can be leveraged towards more efficient algorithm design. In the next section, we will introduce an efficient method for mining all the FCS in a large graph database. Finally, we note that it is possible to generalize the cohesive set with additional connectivity constraints, in a manner similar to [36]. For instance, each induced subgraph may not only be connected, but k-connected as well. Our algorithm can be generalized to handle such constraints as well. However, this is beyond the scope of the present work.

4. MINING FREQUENT COHESIVE SETS

The frequent cohesive set problem is similar to the highly reliable subgraph problem, in that it also lacks the downward closure property (Lemma 1). This creates an algorithmic challenge for the pattern mining process. In order to address this challenge, we develop a novel two-stage mining algorithm. In the first stage, a novel top-down *peeling* process is employed to iteratively refine patterns to make them converge into maximal frequent cohesive sets (MFCS) (Subsections 4.1 and 4.2). In the second stage, we perform a DFS mining process which utilizes the MFCS as the boundary to prune the search space for discovering all the non-maximal frequent cohesive sets (Subsection 4.3). We will now describe these different stages.

4.1 Peeling Algorithm for MFCS

In this subsection, we describe a novel and efficient algorithm for mining the maximal frequent cohesive sets (MFCS). We refer to our approach as the *peeling* algorithm, because we work from *supersets to subsets* during pattern discovery, as patterns are being iteratively refined (or peeled). Clearly, a successful peeling approach requires two main properties: **(a)** We need to dicover the initial patterns containing all the MFCS. **(b)** We need an effective peeling approach in order to converge to the correct MFCS. In the following, we address these two key issues and provide details of the peeling algorithm.

Relaxation Approach: In order to discover the initial patterns which contain all the maximal frequent cohesive sets (MFCS), we relax the *cohesive* condition on a vertex set by allowing connectivity through vertices outside the set. This relaxed problem also turns out to be easier to solve because it satisfies the downward closure property. Specifically, we introduce the *maximal frequent linked sets* (MFLS) problem, which allows connectivity of a vertex set not just through the induced subgraph itself, but other vertices as well.

DEFINITION 4. *(**Maximal Frequent Linked Set Problem**) A subset of vertices $V_s \subseteq V[G]$ in graph G is said to be a* **linked set** *if it belongs to a connected component in G. Given a graph database $D = \{G_1, G_2, \cdots, G_N\}$ with $V(G_1) = V(G_2) = \cdots = V(G_N) = V$ and a minimum support threshold θ, a subset of vertices $V_s \subseteq V$ is referred to as a* **frequent linked set**, *if it is a linked set in at least $\theta \cdot N$ graphs.*

The frequency of V_s being linked in D is denoted as $\widehat{\mathbf{R}}_L[V_s]$. The **maximal frequent linked set** problem tries to identify all the frequent linked sets in D, such that any superset of these sets is not frequently linked. A frequent linked set relaxes the cohesive constraint by not requiring the set to be connected only through the vertices (and edges) of the induced subgraph. It is easy to see that any frequent cohesive set must be a frequent linked set, though the reverse is not necessarily true. Consequently, *if V_s is a maximal frequent cohesive set (MFCS), then there must be a maximal frequent linked set (MFLS) V_s', such that $V_s' \supseteq V_s$*. Thus, the set of all MFLS can naturally serve as the initial pattern set which forms an "upper bound" on all MFCS. Furthermore, unlike the FCS problem, the relaxed MFLS problem has the *down-closure property*, which allows efficient discovery.

Discovering all MFLS is surprisingly simple, because we can reduce it to the classical maximal frequent itemset problem [14], for which many efficient algorithms are available. Specifically, the *transformation* procedure which converts the graph database D into a transactional database T is as follows:
For each graph $G_i \in D$, find all its connected components, i.e., $G_i = C_{i1} \cup C_{i2} \cup \cdots \cup C_{ik}$. Output the vertex set of each connected component as an independent transaction, i.e., $T = T \cup \{V[C_{i1}], V[C_{i2}], \cdots, V[C_{ik}]\}$ ($T = \emptyset$ initially).

LEMMA 4. *The set of all maximal frequent itemsets (MFI) in T with minimum support θ is equivalent to maximal frequent linked sets (MFLS) in D with the same minimum support level.*

Note that the transformation process can be achieved very quickly [20], since it has linear computational complexity with respect to the size of the graph database. Thus, the question is how we can refine these MFLS to produce the final set of all maximal frequent cohesive itemsets (MFCS). Next, we discuss how this initial set can be peeled.

Naïve Peeling: The basic idea of peeling is as follows. For each initially discovered maximal frequent linked set (MFLS) m in D, we keep refining it in order to convert it to a (maximal) frequent cohesive set (FCS). Specifically, if m is not a frequent cohesive set, we first effectively peel the graph database so that it contains only vertices in m. In other words, to refine m, we only work on the partial graph database $D[m] = \{G_1[m], G_2[m], \cdots, G_N[m]\}$, which contains only the induced subgraph of m for each graph in D. Note that once we peel the graphs so that they contain only vertex set m, they may become disconnected (since they are likely to be linked through other vertices in the graph). Then, we can discover all the maximal frequent linked sets (MFLS) on the partial graph database $D[m]$ using the earlier method. Thus, we can

recursively perform this process until all maximal frequent linked sets (MFLS) converge into the frequent cohesive sets.

Algorithm 1 $\mathbf{Peeling}(D, mfls, MFCS)$

Parameter: D: the graph database $D = \{G_1, G_2 \cdots, G_N\}$;
Parameter: $mfls$: the intermediate maximal frequent linked sets;
Parameter: $MFCS$: the final maximal frequent cohesive sets;
1: **for each** $m \in mfls$ **do**
2: **if** $\widehat{\mathbf{R}}[m|D[m]] \geq \theta$ {if m is a frequent cohesive set} **then**
3: **if** $\nexists m' \in MFCS \wedge m' \supseteq m$ **then**
4: $MFCS \leftarrow prune(MFCS \cup \{m\})$; {maximal patterns}
5: **end if**
6: **else**
7: $T \leftarrow transform(D[m])$ {transform graph database to transactional database};
8: $mfls' \leftarrow MFI(T, \theta)$ {maximal frequent itemsets};
9: **Peeling**$(D, mfls', MFCS)$;
10: **end if**
11: **end for**

Algorithm 1 sketches the peeling process. The input is the initial MFLS discovered using the aforementioned MFI mining approach. Then, we invoke the *peeling* algorithm. In the loop (Lines 1 to 13), we iteratively visit each maximal frequent linked set m in $mfls$. If m is indeed a frequent cohesive set ($\widehat{\mathbf{R}}[m|D[m]] \geq \theta$), we try to add it in the result set $MFCS$ (Lines 3 to 5) where the *prune* procedure enforces the maximal constraint. If not, we perform the refinement process described in lines $8-10$. In other words, we discover the maximal frequent linked sets in $D[m]$. We recursively perform peeling for all newly discovered MFLS (Line 11).

THEOREM 2. *(***Correctness of Peeling***) The* **Peeling** *algorithm (Algorithm 1) can discover all maximal frequent cohesive sets from graph database D with minimum support θ.*

The proof of the theorem is omitted for simplicity. Note that the peeling process requires us to recursively invoke the pattern discovery algorithm on smaller sets. It is important to remember that later invocations of the MFI procedure are typically on transactions of *much* smaller length. This is therefore typically very fast at lower levels of the recursion. Furthermore, the cost to compute $\widehat{\mathbf{R}}[m|D[m]]$ and transform a graph database to a transactional database requires a simple DFS scan of $D[m]$ [20], which is a comparatively very small overhead to MFI mining. Nevertheless, the naïve peeling algorithm may redundantly re-examine the same intermediate frequent linked sets (MFLS) multiple times. For example, when two MFLS overlap along a large number of vertices, then their individual peeling processes may produce the same intermediate MFLS. The recursion could then create a combinational explosion, the bulk of which is redundant processing. This needs to be controlled in order to enable more efficient and practical pattern discovery. In the following, we will present a number of elegant methods to avoid duplicate work, and significantly speed up the peeling algorithm.

4.2 Fast Peeling Algorithm

In order to speed up vertex pattern generation, we use the methods of *layered peeling* and *transaction reduction*. We discuss them below.

Layered Peeling: This technique focuses on reducing the processing of redundant intermediate patterns. The basic idea is that instead of performing the recursive peeling of each individual intermediate pattern, we employ a *layered peeling* strategy. For an initial set of patterns in the first layer, we peel them all together to produce another set of intermediate patterns in the second layer. We then perform the same peeling process for the new set of intermediate patterns. If the new patterns (frequent linked sets) are frequent cohesive sets, we remove them from the new layer. More importantly, *each layer is composed of only maximal patterns*. This is based on the following observation:

LEMMA 5. *Given two frequent linked sets m and m', if $m \subseteq m'$, then the maximal frequent cohesive sets (MFCS) contained by m, denoted as $MFCS(m)$, must be contained in $MFCS(m')$. Furthermore, assuming a pattern T is already known to be a FCS, then in the new layer, if it contains an intermediate pattern which is the subset of T, then, the intermediate pattern can be pruned without missing any MFCS.*

This property ensures that it is sufficient to work only with maximal linked sets without losing information. This allows each unique intermediate pattern to be peeled only once when it is necessary for new maximal frequent cohesive set discovery. In addition, for the latter statement, if m is such an intermediate pattern $m \subseteq T$, then m cannot produce any new MFCS. Using our running example in Figure 1 with 100 sampling graphs and $\theta = 0.5$, we need three layers to discover all the MFCS. In the first layer which includes all the initial MFLS in G, we have four: $\{a, b, c, d, f\}$, $\{a, c, d, f, g\}$, $\{c, d, e, f, g\}$, and $\{a, c, d, f, e\}$. Then, in the second layer, we have three MFCS $\{c, d, e, f\}$, $\{a, c, d, f\}$, and $\{a, b\}$, and two intermediate patterns (MFLS) $\{d, e, f, g\}$ and $\{c, d, f, g\}$. Finally, in the third layer, from the last two intermediate patterns, we found one MFCS $\{e, g\}$ and two new intermediate patterns $\{c, d, f\}$ and $\{d, e, f\}$. Since we already knew $\{a, c, d, f\}$ is a MFCS, we can prune $\{c, d, f\}$ from the new layer. Similarly, we can prune $\{d, e, f\}$ because of $\{d, e, f, g\}$. Thus, the third layer becomes empty and we already have found all the MFLS.

Transaction Reduction: In the method discussed above, we ensure that patterns in a given layer do not contain one another. However, this does not account for the fact that patterns which are produced in lower layers may be subsets of those produced in earlier layers. Therefore, in order to further speed up the peeling algorithm, we would like to prevent such non-maximal patterns from being generated in the first place. The following lemma provides an important tool for achieving this.

LEMMA 6. *Let us assume that the patterns in the current layer L are visited sequentially. Let $P \subseteq L$ include those patterns which have already been visited (peeled) at any given time point. Let V_{ij} be the vertex set of a connected component C_{ij} in $G_i[m]$, where $m \in L$ has not been peeled. If there is another pattern (frequent linked sets) m' which has been processed (peeled) ($m' \in P$) and $V_{ij} \subseteq m$, then we can safely drop V_{ij} in the transactional database transformed from $D[m]$ without losing any potential maximal frequent cohesive sets in D.*

The same dropping condition can be applied for any discovered (maximal) frequent cohesive sets in D. If any transaction is a subset of a (maximal) frequent cohesive set in D, we can also safely drop it. This is because transactions such as V_{ij} cannot contribute to any new MFCS, and can only help generate patterns which are subsets of patterns which have already been visited. Thus, dropping such transactions in the first place can help prevent the generation of unnecessary patterns. In addition, such transaction reduction can also help reduce the computational cost, because it results in a smaller database for MFI.

Overall Algorithm: Algorithm 2 depicts the fast peeling process. Initially, L contains all the maximal frequent linked sets in D (Line

1). Then we iteratively visit each element m of L in decreasing order of pattern size. This order is chosen in order to maximize the efficiency of search space pruning. This is based on Lemma 6. We try to add those frequent cohesive patterns into the result set $MFCS$ (line 6 to 10), where *prune* maintains the maximal constraint for $MFCS$. For those that are not frequent cohesive sets, *transReduc* produces and reduces the transactions which are contained by patterns in P or $MFCS$ (Line 11) according to Lemma 6. This reduced database T is used to generate the intermediate patterns $mfls'$ (Line 13). These are merged into L' by the *prune* procedure to maintain the maximal constraint(Line 14). Finally, the new layer will be processed (Line 17) until there are no new patterns to be generated.

Algorithm 2 $\mathbf{FastPeeling}(D)$)

Parameter: D: the graph database $D = \{G_1, G_2 \cdots, G_N\}$; {Step 1: generating initial patterns}
1: $L \leftarrow MFI(transform[D], \theta)$ {L: existing layer};
2: $L' \leftarrow \emptyset$ {L': new layer}; $MFCS \leftarrow \emptyset$ {the result set};
3: **while** $L \neq \emptyset$ **do**
4: $P \leftarrow \emptyset$ {already peeled patterns in L}
5: **for each** $m \in L$ {decreasing order of the pattern size} **do**
6: **if** $\widehat{\mathbf{R}}[m|D[m]] \geq \theta$ {if m is a frequent cohesive set} **then**
7: **if** $\nexists m' \in MFCS \wedge m' \supseteq m$ **then**
8: $MFCS \leftarrow prune(MFCS \cup \{m\})$;
9: **end if**
10: **else**
11: $T \leftarrow \mathbf{transReduce}(D[m], P \cup MFCS)$ {Lemma 6}
12: $mfls' \leftarrow MFI(T, \theta)$ {maximal frequent itemsets};
13: $L' \leftarrow prune(L' \cup mfls')$; {Lemma 5: maximal patterns}
14: **end if**
15: $P \leftarrow P \cup \{m\}$;
16: **end for**
17: $L \leftarrow prune(L' \cup MFCS) \setminus MFCS$ {Lemma 5: pruning using FCS}; $L' \leftarrow \emptyset$;
18: **end while**

The computational complexity of our algorithm is dominated by the cost of mining Maximal Frequent Itemsets (MFI). Let the cost of mining MFI in the transformed transactional database from the entire graph database D be $O(MFI_D)$ (Line 1). For i-th layer L, we can break it into the minimal number c_i of batches, such that the patterns are all disjoint with one another in each batch. Thus, we can see the overall cost of each layer is bounded by $O(c_i MFI_D)$. If there are a total of k layers, then the total computational complexity is $O((\sum_{i=1}^{k} c_i)MFI_D)$. However, since the transactional database in each batch is typically much smaller than the first transactional database (Line 1), the total cost of mining MFI for all layers is generally even smaller than the cost of MFI mining once at the start (Line 1). We note the number of total layers is typically quite small in practice. For example, in all our experiments, this number was less than 5. Therefore, the total computational complexity of the peeling approach is proportional to a single execution of MFI_D (or $O(MFI_D)$). As we will show from the experimental results in Section 5, the overall computational time of peeling is no higher than 2 times that of mining MFI in the first round.

4.3 DFS Mining for Non-Maximal FCS

While the previous section provides a way to discover maximal patterns, we also need to discover all non-maximal frequent cohesive sets. A naïve approach would be to directly apply the peeling algorithm to further discover the remaining sets. To do that, we can explicitly remove each vertex from the (maximal) frequent cohesive set to produce the intermediate patterns for further peeling. For instance, for a (maximal) pattern $\{a, b, c, d\}$, we can generate new intermediate patterns $\{b, c, d\}$, $\{a, c, d\}$, $\{a, b, d\}$, and $\{a, b, c\}$. Clearly, this may significantly increase the number of times we need to scan the graphs, and invoke the maximal frequent itemset mining algorithm. The speedup techniques of Lemma 5 and 6 are also not directly applicable in this case. The key question is whether we can leverage our knowledge of the maximal frequent cohesive sets in order to enable discovery of the non-maximal ones. In the following, we provide a positive answer to this question by amortizing the discovery across different maximal patterns.

The basic idea of our mining algorithm is that we perform a DFS traversal on $\mathcal{G}$ to enumerate any *connected vertex sets* in V which are contained by at least one MFCS being covered in the peeling algorithm. Then, for each of these vertex sets, we perform an efficient test to determine whether they are frequently cohesive. Since there is a standard method for enumerating connected vertex sets along the lines of enumerating cliques [23, 8], the main issue is how we can efficiently control the enumeration boundary. Clearly, we may simply test whether each discovered vertex set is contained by a MFCS, but such a test can be rather costly [37]. Therefore, we next discuss how to speed up this test.

Fast Subset Checking: Each MFCS is assigned a unique ID, and each vertex in the uncertain graph is associated with a list that records the IDs of each MFCS containing it. Furthermore, each enumerating vertex set maintains a list which is the intersection of the lists from its individual vertices. In other words, the list records all the MFCS which contain the vertex set. Importantly, this list can be maintained in an incremental fashion. We also note that when the list for a vertex set is empty, it is implied that no MFCS contains it.

Fast Connectivity Test: In order to test whether a vertex set V_s is frequently cohesive, the straightforward method would be to directly check the connectivity of each induced subgraph $G_1[V_s]$, $G_2[V_s]$, $\cdots$, $G_N[V_s]$. An observation which enables convenient speedup is as follows: *For a fully connected induced subgraph, if a new vertex is added and it is adjacent to at least one vertex in the subgraph, then the new subgraph is also connected.*

This implies that we can use a binary vector to record whether each induced subgraph of the current vertex set is connected, and then we can apply it to test whether its immediate expansion by one vertex is connected. It is only when the induced subgraph is not fully connected that we need to traverse its immediate expansion in order to determine its connectivity.

Algorithm Description: Algorithm 3 describes a DFS mining procedure in order to discover all non-maximal frequent cohesive sets. First, we note that parameters v, V_s, N and Ex are the standard parameters for enumerating the connected vertex sets in a graph [23, 8]: v is the newly expanded vertex in the current connected vertex set V_s in uncertain graph $\mathcal{G}$; N records all the neighbors of V_s, which can be possibly added into V_s; and Ex is the exclusion list of vertices which should not be added into V_s (a basic mechanism to avoid the redundant enumeration of the same connected vertex set in a graph). Ex typically records those vertices visited earlier in the DFS enumeration order. The parameter ID is for the aforementioned boundary test and vector Con is for the fast connectivity test.

For each newly expanded vertex set V_s (to which vertex v was newly added), Algorithm 3 first checks whether it is a FCS using the *fast connectivity test* (Lines 1 to 10). Then, no matter whether the vertex set is a FCS or not, each of its neighbors except those in the exclusion list (avoid redundant enumeration) will be visited (Line 11). This is a typical way for recursively enumerating the connected vertex set. Specifically, ID' is the list recording all the IDs of those MFCS containing $V_s \cup \{w\}$ and it is maintained incrementally (Line 12). Once a vertex w is visited, we put it immedi-

Algorithm 3 $\mathbf{MiningNonMaximal}(v, V_s, ID, Con, N, Ex)$

Parameter: v {the newly added edge-vertex in V_s}
Parameter: V_s {the current vertex sets}
Parameter: ID {the IDs of maximal cohesive sets containing V_s}
Parameter: Con {the binary vector for connectivity}
Parameter: N {the neighbors of V_s}
Parameter: Ex {exclusion list of vertices (already expanded)}
1: **for each** $G_i \in D$ **do**
2: **if** $Con[i]$ **then**
3: $Con'[i] \leftarrow (Neighbor(v|G_i) \cap V_s) \neq \emptyset$;
4: **else**
5: $Con'[i] \leftarrow Connected(G_i[V_s])$;
6: **end if**
7: **end for**
8: **if** $\widehat{\mathbf{R}}[V_s] \geq \theta$ {using Con' to compute} **then**
9: $FCS \leftarrow FCS \cup \{V_s\}$ {non-maximal frequent cohesive sets}
10: **end if**
11: **for each** $w \in N \backslash Ex$ **do**
12: $ID' \leftarrow ID \cap ID[w]$; {MFCS containing V_s}
13: $Ex \leftarrow Ex \cup \{w\}$;
14: **if** $|ID'| > 1 \vee (|ID'| = 1 \wedge V_s \cup \{v\} \neq$ MFCS in $ID')$ **then**
15: **MiningNonMaximal**$(w, V_s \cup \{v\}, ID', Con', Neighbor(w|\mathcal{G}) \cup N, Ex)$;
16: **end if**
17: **end for**

Procedure Main
1: $Ex \leftarrow \emptyset$;
2: **for each** $w \in V \backslash Ex$ **do**
3: $Ex \leftarrow Ex \cup \{w\}$;
4: **MiningNonMaximal**$(w, \{w\}, ID[w], \mathbf{0}, \mathbf{Neighbor(w|\mathcal{G})}, \mathbf{Ex})$;
5: **end for**

ately in the exclusion list so that the latter iteration will not visit it again (Line 13). For each newly expanded vertex set, when either its list ID' is contained by at least two MFCS or is a strict subset of the only MFCS ($|ID'| = 1$), we know it has not reached the boundary yet and cannot be considered a candidate for non-maximal FCS (Lines $14 - 15$).

Note that the worst case computational complexity of this algorithm is determined by the total number of connected vertex subsets which are bounded by MFCS. For each connected vertex set, we need to traverse each of its induced subgraphs in the graph database. This has linear cost.

5. EXPERIMENTAL EVALUATION

In this section, we present experimental results studying the accuracy and efficiency of our method. Specifically, we are interested in the following two questions:

1. Accuracy: How well does the sampling approach approximate the set of highly reliable subgraphs S_α? Recall that we utilize two sets $\overline{S}$ and $\underline{S}$ ($\overline{S} \supseteq \underline{S}$) to approximate S_α, and these two sets are designed to measure recall and precision respectively. When the two sets are similar, they can provide accurate estimation of the S_α according to Theorem 1. In such cases, the precision and recall are both high. It is easy to see that the fraction $\beta = \frac{|\underline{S}|}{|\overline{S}|}$ provides a good indicator for the accuracy of the sampling approach for experimental evaluation.

2. Efficiency: What is the performance in terms of overall running time? As we mentioned before, the sampling approach consists of two steps for (1) sampling dataset D_1 and for (2) sampling dataset D_2.

In step (1), we need to discover all frequent cohesive sets (FCS) from D_1. Specifically, step (1) contains two stages: in the first stage, we apply peeling (naïve or fast) algorithms to discover all maximal frequent cohesive set (MFCS); and in the second stage, we utilize a DFS mining process using MFCS to discover all the remaining Non-Maximal FCS. Given this, we would like to understand how the computational time is distributed between Steps (1) and (2), and a quantification of the efficiency advantage of fast peeling over the naïve approach. Step (2) can be implemented much more efficiently, because it requires us to only check membership on D_2.

All algorithms were implemented using C++ and the Standard Template Library (STL), and were conducted on a 2.0GHz Dual Core AMD Opteron CPU with 4.0GB RAM running Linux.

Table 1: β(%) vs varying ϵ ($\delta = 0.01, \alpha = 0.99$)

	ϵ =0.030	ϵ =0.035	ϵ =0.040	ϵ =0.045	ϵ =0.050
Yeast	55.92	54.26	53.63	53.55	49.54
Fly	57.86	56.90	52.40	52.57	52.62
Mouse	98.65	99.10	99.10	99.10	99.10
Rat	100.00	100.00	100.00	100.00	100.00
DBLP	73.90	70.28	70.01	68.60	65.11

Table 2: β(%) vs varying δ ($\epsilon = 0.05, \alpha = 0.99$)

	δ=0.01	δ=0.008	δ=0.006	δ=0.004	δ=0.002
Yeast	46.78	46.72	49.66	49.30	48.76
Fly	50.76	51.46	51.46	52.23	52.91
Mouse	99.10	99.10	99.10	99.10	99.10
Rat	100.00	100.00	100.00	100.00	100.00
DBLP	65.49	66.73	67.42	65.32	66.23

5.1 Experimental Results on Real Datasets

In this subsection, we report our experimental results on five real datasets: four protein-protein interaction (PPI) uncertain graphs and one coauthor graph. The PPI datasets are integrated from BioGRID database and STRING databases, and are provided by the author in [39]. The coauthorship network is derived from DBLP, and is extracted from the dataset provided by the author in [30]. The summary of those datasets are listed in Figure 2, where the last column $avg(p_e)$ indicates the average edge probability in the uncertain graph.

In the first three groups of experiments, we focus on studying how the accuracy indicator β is affected by the three user-defined parameters, the reliability threshold α, the confidence level δ, and ϵ which directly relates to β (Section 3).

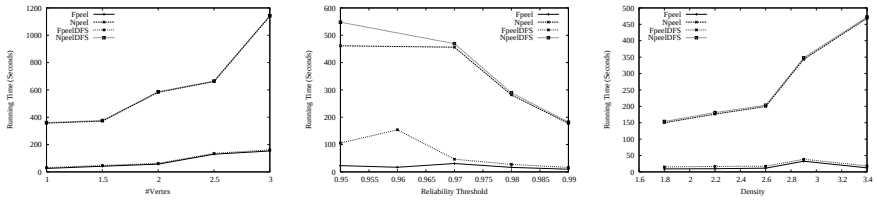
Varying ϵ: In this experiment, we fix the reliability threshold α and the confidence level δ with $\alpha = 0.99$ and $\delta = 0.01$ (99% confidence), and we vary the ϵ from 0.03 to 0.05. Table 1 reports the variation of the accuracy indicator β with ϵ on each dataset. First, it is evident that β is always at least 50%, and in 3 out of 5 datasets (Mouse, Rat, and DBLP), β is generally much higher than 65%. The lowest values of β were obtained for the Yeast and Fly data sets, but were still above or very close to 50%. Taking $\overline{S}$ as the example, this simply suggests that at least half of the discovered subgraphs in $\overline{S}$ are guaranteed to be highly reliable ($\geq \alpha = 0.99$), and the expected fraction of HRS being missed by $\overline{S}$ is no more than 1%. The best performance was obtained on Mouse and Rat, for which β was very close to 1. A detailed analysis shows that most of the subgraphs discovered from these two datasets are very small and their number is also small. They also seem to form a

Table 3: β(%) vs varying α ($\epsilon = 0.05, \delta = 0.01$)

	α=0.99	α=0.97	α=0.95	α=0.93	α=0.90
Yeast	48.64	9.25	5.25	52.94	80.92
Fly	51.27	41.87	30.92	52.77	63.69
Mouse	99.10	92.36	86.42	94.24	96.04
Rat	100.00	100.00	100.00	98.00	97.13
DBLP	65.51	53.82	45.64	68.13	83.84

	$\|V\|$	$\|E\|$	$agv(p_e)$
Yeast	162	300	0.148
Fly	3751	7384	0.456
Mouse	199	286	0.413
Rat	130	178	0.374
DBLP	1000	2356	0.560

Figure 2: datasets summary



(a) Time vs. $|V|$ (b) Time vs. α (c) Time vs. Density

Figure 3: Running Time vs. Random Block Model Graphs

Table 4: Execution time(seconds) for real datasets ($\epsilon = 0.05, \delta = 0.01, \alpha = 0.99$)

	NpeelDFS			FpeelDFS		
	Step 1-p	Step 1-d	Step 2	Step 1-p	Step 1-d	Step 2
Yeast	30.03	3.05	47.23	0.99	2.88	44.71
Fly	437.08	22.76	209.90	23.28	19.22	180.95
Mouse	28.79	0.05	0.86	0.75	0.05	0.84
Rat	8.67	0.02	0.44	0.39	0.02	0.40
DBLP	368.67	63.60	26.64	26.58	61.77	28.52

few small components. Thus, these small components are highly reliable and are easily discovered without too many false positives. Second, it is evident that β increases for lower values of ϵ. This is because reduced values of ϵ lead to higher sample sizes, and also because smaller ϵ would result in more accurate thresholds for the hypothesis test. In other words, $\alpha-\epsilon$ increases and $\alpha+\epsilon$ decreases. Thus, $\overline{S}$ tends to reduce and $\underline{S}$ tends to increase as ϵ reduces.

Varying δ: In this experiment, we fix the reliability threshold α and parameter ϵ ($\alpha = 0.99$ and $\epsilon = 0.05$), and we vary δ from 0.01 to 0.002, which correspond to very high levels of confidence. Table 2 reports the accuracy indicator β with respect to different values of δ on each dataset. First, we can see that the overall accuracy indicator β is consistent with the first experiment, because it is usually larger than 50%. Second, we observe that as δ decreases (confidence level $1-\delta$ increases), β also tends to increase. However, the margin of increase is relatively small. This is because the influence of δ on sample size is proportional to $\ln(1/\delta)$, whereas the influence of ϵ on sample size is proportional to $1/\epsilon^2$. Therefore, δ has a much smaller influence on the sample size. This also suggests that the overall recall rate of $\overline{S}$ is quite consistent, because of our earlier observation that the expected false negative rate is δ.

Varying α: In this experiment, we fix the confidence parameter δ and parameter ϵ ($\delta = 0.01$ and $\epsilon = 0.05$), and we vary the reliability threshold α from 0.99 to 0.90. Table 3 reports the accuracy indicator β with respect to the different α on each dataset. Interestingly, we note that when α reduces, β first decreases and then increases. This reduction in β is particularly noticeable for the Yeast data set, though it is moderate for all other data sets. The reason for the decreasing-increasing trend seems related to the distribution of highly reliable subgraphs. When α is very high ($\alpha = 0.99$), both $\underline{S}$ and $\overline{S}$ are quite small and also close. As α slightly decreases, $\overline{S}$ (defined by threshold $\alpha+\epsilon$) remains relatively stable, but $\overline{S}$ (with threshold $\alpha-\epsilon$) can grow rather quickly. This results in a decrease in β. However, when α further decreases, the set $\overline{S}$ grows faster than $\underline{S}$. Therefore, the difference between them becomes smaller.

Execution Time: In this experiment, we study the computational time of Steps 1 and 2, and compare the performance of naïve peeling and fast peeling. Specifically, we fix the parameters α, δ and ϵ, and report the peeling time (discovering MFCS in Step 1), the DFS mining time (discovering Non-maximal FCS in Step 1), and the time in Step 2. We denote NpeelDFS to be the algorithm utilizing the naïve peeling (Algorithm 1) and FpeelDFS (Algorithm 2) to the algorithm utilizing the fast peeling. We can see that in most of the cases, the peeling stage requires most of the computational time in NpeelDFS. The FpeelDFS algorithm is faster than the naïve peeling approach by more than one order of magnitude. Furthermore, we observe that the overall computational times of Steps 1 and Step 2 become comparable in the case of the fast peeling approach. Since the processing in Step 2 is rather straightforward (checking whether a given subgraph is a FCS on the larger dataset D_2), we do not consider further optimization of this step here.

5.2 Experimental Results on Synthetic Datasets

Here, we focus on studying the running time of our mining approach on synthetic datasets. Specifically, we utilize the block-random graph model [24], which can generate both the Erdös-Rényi random graph and Scale-free random graph, along with a specified community structure. The edge existence probability is uniformly generated between 0 and 1. We report the overall running time of NpeelDFS and FpeelDFS, and their respective peeling time (Npeel and Fpeel) in Step 1. The default parameters are $\alpha = 0.99$, $\delta = 0.01$, and $\epsilon = 0.05$. Figure 3(a) reports the running time with respect to the graph size as the the number of vertices change from 1000 to 3000 with the average edge density fixed at 1.5. Figure 3(b) reports the running time with respect to the reliability threshold varying from 0.95 to 0.99 on an uncertain graph with 1000 nodes and edge density fixed to 1.5. Figure 3(c) reports the running time with respect to the edge density varying from 1.8 to 3.5 on an uncertain graph with 1000 nodes. Here, we can see that throughout these experiments, the overall running time of the fast peeling based approach FpeelDFS is much faster than that of the naïve peeling approach NpeelDFS. In addition, in most of the cases, it seems the peeling time (discovering MFCS) is also a major component of the overall running time.

6. RELATED WORK

The work closest to ours is the most reliable subgraph problem [16, 17, 25, 18]. Given a set of vertices, this problem tries to remove K edges from the original graph so that the remaining subgraph can maximize the probability of these vertices belonging to one connected component. Thus, the highly reliable subgraph (HRS) problem can be viewed as a generalization of the most reliable subgraph problem, because no initial set of vertices is specified. Furthermore, HRS also puts more constraints on vertex set reliability, because it requires all vertices in each subgraph to be fully connected, whereas the most reliable problem only requires the targeted set of vertices in the subgraph to be connected. Because of these differences, the methods developed for most reliable subgraph mining cannot be generalized to this new problem.

Mining uncertain graphs has recently attracted much attention in the data mining and database research communities [30, 38, 39, 40]. Specifically, Zou *et al.* study mining frequent subgraphs [39] and top k-cliques [40] in a single uncertain graph. Potamias *et*

al. study the k-Nearest Neighbor problem in uncertain graphs [30]. Yuan *et al.* study a new variant of the shortest path problem in an uncertain graph [38], and Jin *et al.* study the distance-constraint reachability problem, a generalization of the classic two-terminal reliability problem [22].

The frequent cohesive set (FCS) discovery problem studied in this work is closely related to frequent pattern mining [13]. This broad subfield has been extensively studied since its inception in the early nineties. However, the frequent cohesive set (FCS) problem has not been studied before. Furthermore, we deviate from natural pattern mining approaches which focus on bottom-up strategies (level-wise or pattern-growth). Here, we introduce a novel peeling approach which enables top-down pattern discovery.

7. CONCLUSIONS AND SUMMARY

In this paper, we present a method for mining reliable subgraph patterns in uncertain graphs. Such problems are extremely challenging in the uncertain scenario and tend to be $\#P$-complete. We present a probabilistic method for mining such reliable graphs which retains efficiency and also provides probabilistic bounds for accuracy. A novel peeling approach reduces the computational complexity by carefully pruning of large portions of the massive search space during the pattern discovery and exploration process. We present experimental results illustrating the effectiveness and efficiency of the method.

8. REFERENCES

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

[2] C. C. Aggarwal, editor. *Managing and Mining Uncertain Data*. Advances in Database Systems. Springer, 2009.

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

[4] J. S. Bader, A. Chaudhuri, J. M. Rothberg, and J. Chant. Gaining confidence in high-throughput protein interaction networks. *Nature Biotechnology*, 22(1):78–85, December 2003.

[5] M. O. Ball. Computational complexity of network reliability analysis: An overview. *IEEE Transactions on Reliability*, 35:230–239, 1986.

[6] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

[7] Y. Benjamini and D. Yekutieli. The Control of the False Discovery Rate in Multiple Testing under Dependency. *The Annals of Statistics*, 29(4):1165–1188, 2001.

[8] C. Chen, X. Yan, F. Zhu, and J. Han. gapprox: Mining frequent approximate patterns from a massive network. In *ICDM*, pages 445–450, 2007.

[9] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[10] C. J. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, Inc., 1987.

[11] 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'07*, pages 1721–1729, 2007.

[12] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *WWW'04*, pages 403–412, 2004.

[13] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent Pattern Mining: Current Status and Future Directions. *Data Mining and Knowledge Discovery*, 14(1), 2007.

[14] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 2nd edition, 2006.

[15] S. Hanhijärvi, K. Puolamäki, and G. C. Garriga. Multiple hypothesis testing in pattern discovery, 2009. arXiv:0906.5263v1 [stat.ML].

[16] P. Hintsanen. The most reliable subgraph problem. In *PKDD*, pages 471–478, 2007.

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

[18] P. Hintsanen, H. Toivonen, and P. Sevon. Fast discovery of reliable subnetworks. In *ASONAM*, pages 104–111, 2010.

[19] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[20] J. Hopcroft and R. Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Commun. ACM*, 16:372–378, June 1973.

[21] R. Jiang, Z. Tu, T. Chen, and F. Sun. Network motif identification in stochastic networks. *PNAS*, 103(25):9404–9409, June 2006.

[22] R. Jin, L. Liu, B. Ding, and H. Wang. Distance-constraint reachability computation in uncertain graphs. In *Proceedings of the VLDB Endowment*, volume 4, 2011.

[23] R. Jin, S. McCallen, and E. Almaas. Trend motif: A graph mining approach for analysis of dynamic complex networks. In *ICDM*, pages 541–546, 2007.

[24] B. Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Phys. Rev. E*, 83(1):016107, Jan 2011.

[25] M. Kasari, H. Toivonen, and P. Hintsanen. Fast discovery of reliable *k*-terminal subgraphs. In *PAKDD (2)*, pages 168–177, 2010.

[26] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

[27] A. Kirsch, M. Mitzenmacher, A. Pietracaprina, G. Pucci, E. Upfal, and F. Vandin. An efficient rigorous approach for identifying statistically significant frequent itemsets. In *PODS'09*, pages 117–126, 2009.

[28] V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal. A survey of algorithms for dense subgraph discovery. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, pages 303–336. Springer US, 2010.

[29] V. Manfredi, R. Hancock, and J. Kurose. Robust routing in dynamic manets. In *Annual Conference of the International Technology Alliance (ACITA)*, 2008.

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

[31] G. Rubino. Network reliability evaluation. In *Network performance modeling and simulation*, pages 275–302. 1999.

[32] M. Stoer and F. Wagner. A simple min-cut algorithm. *J. ACM*, 44:585–591, July 1997.

[33] G. Swamynathan, C. Wilson, B. Boe, K. Almeroth, and B. Y. Zhao. Do social networks improve e-commerce?: a study on social marketplaces. In *WOSP '08: Proceedings of the first workshop on Online social networks*, pages 1–6, 2008.

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

[35] D. R. White and F. Harary. The cohesiveness of blocks in social networks: Node connectivity and conditional density. *Sociological Methodology*, 31:305–359, 2001.

[36] X. Yan, X. J. Zhou, and J. Han. Mining closed relational graphs with connectivity constraints. In *KDD '05*, 2005.

[37] D. M. Yellin. An algorithm for dynamic subset and intersection testing. *Theoretical Computer Science*, 129(2):397–406, 1994.

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

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

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

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