Efficient Mining of Frequent Patterns on Uncertain Graphs

Yifan Chen, Xiang Zhao, *Member, IEEE*, Xuemin Lin, *Fellow, IEEE*, Yang Wang, and Deke Guo, *Senior Member, IEEE*

Abstract—Uncertainty is intrinsic to a wide spectrum of real-life applications, which inevitably applies to graph data. Representative uncertain graphs are seen in bio-informatics, social networks, etc. This paper motivates the problem of frequent subgraph mining on single uncertain graphs, and investigates two different - probabilistic and expected - semantics in terms of support definitions. First, we present an enumeration-evaluation algorithm to solve the problem under probabilistic semantics. By showing the support computation under probabilistic semantics is #P-complete, we develop an approximation algorithm with accuracy guarantee for efficient problem-solving. To enhance the solution, we devise computation sharing techniques to achieve better mining performance. Afterwards, the algorithm is extended in a similar flavor to handle the problem under expected semantics, where checkpoint-based pruning and validation techniques are integrated. Experiment results on real-life datasets confirm the practical usability of the mining algorithms.

Index Terms—Uncertain graphs, frequent subgraphs, approximation algorithms, pattern mining

1 Introduction

Uncertainty is intrinsic to a wide spectrum of real-life applications, either endogenous or extraneous. For example, in a cooperation social network, given people `Bill` and `Matthew`, it may not be possible to definitely assert a relation of the form "`Bill` cooperates well with `Matthew`", using available information at hand. Our confidence in such relation is commonly quantified by probability. We say that the relation exists with a probability of p, and the value of p is determined manually by domain experts using available information, or automatically by information extraction and generation rules. Nowadays, the needs for managing uncertainty become more noteworthy in the big data era, where data of various quality largely emerge. In this paper, thus, we focus on *uncertain graphs*, where our knowledge is presented as a graph with uncertainty associated to edges. Besides aforementioned social networks [1], [2], uncertain graph model is adopted in communication networks [3], [4], wireless sensor networks [5], protein interaction networks [6], [7], biological regulatory networks [8], etc.

Frequent pattern mining has been a focused theme in data mining for more than a decade, making remarkable progress. Graph patterns, or frequent subgraphs, are of particular interest lately, which are subgraphs found from a *collection of small graphs* or *single large graph* with support no less than a user-specified threshold. Frequent subgraphs are useful at characterizing graph datasets, classifying and clustering graphs, and building structural indices [9].

While the notion of frequent subgraph and methods for mining frequent subgraphs on deterministic graphs are well understood, the case becomes more intriguing and less studied on uncertain graphs. An uncertain graph is a special edge-weighted graph, where the weight on each edge (u, v) is the existence probability of the connection between vertices u and v. Lately, research effort has been dedicated to frequent subgraph mining (FSM) on a collection of small uncertain graphs. Being equally important, however, the problem on single large uncertain graphs remains open, given that real-life large networks are increasingly involved with uncertainty in nature.

Motivations. We motivate the problem from three different application domains.

Example 1. In modern biology, biologists are interested in identifying functional modules and evolutionarily conserved subnetworks from protein-protein interaction (PPI) networks. Recent study proposed to discover patterns of functional associations in PPI for the purpose of function prediction [10]. Functions that a protein performs are assigned to the corresponding node as a label. A functional association pattern is then a labeled subgraph which occurs frequently in the network. Frequent subgraph mining is naturally applied to this task, which has achieved better accuracy [10], in comparison with conventional methods. In practice, PPI datasets, such as STRING [1] and BioGRID,[2] are endowed with uncertain

• Y. Chen and D. Guo are with the National University of Defense Technology, Changsha 410073, China. E-mail: yfchen.nudt.edu.cn, dekeguo@nudt.edu.cn.
• X. Zhao is with the National University of Defense Technology, Changsha 410073, China, and the Collaborative Innovation Center of Geospatial Technology, Wuhan 430079, China. E-mail: xiangzhao@nudt.edu.cn.
• X. Lin is with The University of New South Wales, Sydney 2052, Australia. E-mail: lxue@cse.unsw.edu.au.
• Y. Wang is with the Dalian University of Technology, Dalian 116023, China, and The University of New South Wales, Sydney 2052, Australia. E-mail: wangy@cse.unsw.edu.au.

Manuscript received 13 Apr. 2016; revised 14 Feb. 2018; accepted 21 Apr. 2018. Date of publication 26 Apr. 2018; date of current version 9 Jan. 2019.
(Corresponding authors: Xiang Zhao and Deke Guo.)
Recommended for acceptance by R. Cheng.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2018.2830336

1. https://string-db.org
2. https://thebiogrid.org

pairwise interactions, which are suggested based on experimental statistics [6]. In this connection, we are motivated to discover subgraphs that frequently appear in an uncertain graph with high probabilities, which is the basis for protein function prediction.

Example 2. The example arises from knowledge-based document modeling. Specifically, we use a *semantic graph* to express various concepts [11], where entities are regarded as nodes, and relations as edges labeled with different categories; moreover, edges capture the *degree of associativity (relevance)* between entities. It is seen that semantic graph pertinently corresponds to our abstract uncertain graph model, where associativity is reflected by probability. Over it, similarity of two documents can be semantically evaluated by comparing their *document graphs* [11], which are subgraphs of the semantic graph. The extraction and comparison of document graphs involve matching subgraphs, which can be fairly computationally expensive [12], [13]. To facilitate pattern matching, frequent subgraphs are generally mined and indexed [14]. Thus, mining frequent subgraphs from uncertain graphs fills in the gap to build indices for matching operations on the uncertain setting.

Example 3. We are also motivated from social networks, e.g., the extracted co-authorship network originated from DBLP containing researchers of computer sciences. This co-authorship network has widely been modeled as uncertain graph [15], [16], [17]. For DBLP, two authors are linked if they have coauthored a journal or a conference paper. The number of papers coauthored is recorded, with which the likelihood of future re-activation of the collaboration can be evaluated, which is a probability in nature. While efficient and intelligent tools are required to analyze the uncertain social networks, frequent subgraph mining is apparently one of it, which is able to reveal strong collaboration tendency in probabilities.

While the significance of a subgraph pattern on a deterministic graph is measured by *support*, this notion does not make sense on uncertain graphs, since the containment relationship becomes vague, or non-deterministic, due to the probabilistic presence of graph structure.

Existing work defined support on a collection of small uncertain graphs [18], [19], which counts the contribution from an implied graph as long as it contains the subgraph *once*. Two different definitions are given for support, which are under *expected semantics* and *probabilistic semantics*, respectively. FSM under expected semantics is preferable to exploring motifs while that under probabilistic semantics is more suitable for detecting features. Inspired by these efforts, we extend the notions, and also define support on single large uncertain graphs under the two semantics - expected and probabilistic. For support under probabilistic semantics, it is defined as the aggregated probabilities of the implied graphs having support larger than a threshold, which is a *survival function* of the support. For support under expected semantics, it is defined as the aggregated contribution to support, weighted to its existence probability over all implied graphs, e.g., a *probability distribution* over the support in all implied graphs of the uncertain graph. Then, subgraphs surpassing a given support threshold are considered *frequent*. Due to the *shift of definition*, existing algorithms on a collection of uncertain graphs are no longer applicable to single uncertain graphs. Therefore, we propose efficient solutions with accuracy guarantee, where the *computational challenge* of proper handling edge-based probability and vertex-based support is also addressed.

Contributions. To the best of our knowledge, this work makes a first effort on FSM on single uncertain graphs. We inquire into the notion of frequent subgraphs on single uncertain graphs, hardness of support computation, and solution to efficient mining. On top of the preliminary version in [20], in this article we make substantial improvements:

(1) Besides expected semantics, to discover subgraphs that are frequent to exist in reality with a significantly high chance, we define probabilistically frequent subgraphs on single uncertain graphs, which captures the features of majority possible graphs implied thereby.
(2) We enhance the computational sharing technique with a formal cost analysis. By showing cost minimization is NP-hard, we propose to heuristically integrate it with the computation sharing tree, which achieves $O(1)$ difference on average in regard to approximation quality.
(3) As to the check-point mechanism, we formally investigate the optimal choice of checkpoints. To seek additional performance gain, besides upper bound, we also put forward a lower bound by aggregating only the support contribution from unique sample graphs.
(4) We conduct an extensive experimental study of the proposed methods and techniques. A case study is provided to showcase the significance of the problem. Extension to more general uncertain graphs with uncertainty existing on both vertices and edges is also discussed.

Organization. Two problems are formulated in Section 2. Section 3 presents a framework that embraces both problems. We investigate support evaluation, respectively, under probabilistic and expected semantics in Sections 4 and 5. Section 6 describes the experiments. We discuss related work in Section 7, and conclude the paper in Section 8.

2 Preliminary

We define the problem of FSM on deterministic graphs, and then generalize to uncertain graphs.

2.1 Deterministic Graph

A *deterministic graph* G is a tuple $(V_G, E_G, l_V, l_E, \Sigma)$, where V_G is a set of vertices, $E_G \subseteq V_G \times V_G$ is a set of edges, and $l_V : V_G \rightarrow \Sigma$, $l_E : E_G \rightarrow \Sigma$ are the labeling functions that assign labels to vertices and edges. $|V_G|$ and $|E_G|$ are the numbers of vertices and edges in G, respectively. Without loss of generality, we consider undirected *simple* graphs.

A graph g is *subgraph isomorphic* to another graph G, denoted by $g \sqsubseteq G$, if there exists an *injection* $f : V_g \rightarrow V_G$ such that (1) $\forall v \in V_g,\ \ f(v) \in V_G \wedge l_g(v) = l_G(f(v))$; and (2) $\forall (u,v) \in E_g$, $(f(u), f(v)) \in E_G \wedge l_g(u,v) = l_G(f(u), f(v))$. Hence, g is a *subgraph* of G, G is a *supergraph* of g, and $f(g)$ is an *embedding* of g in G. g is a *direct supergraph* of g' if $g' \sqsubseteq g$ and $|E_g| = |E_{g'}| + 1$.

Consider two graphs $g \sqsubseteq G$, and a *support threshold* τ, and assume that there is a function to measure the *support* of g in

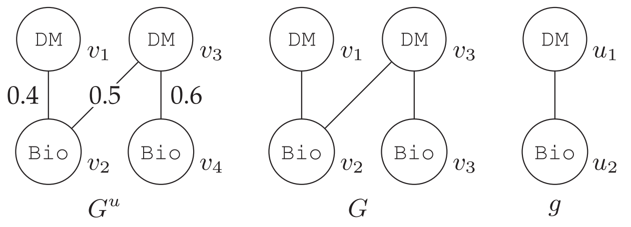


Fig. 1. Example graphs.

G, denoted by $sup(g, G)$. If $sup(g, G) \geq \tau$, we say g is a *frequent subgraph*. Based on the discussion in Appendix C.1, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2018.2830336 we adopt minimum image based support in this paper.

Consider a set of distinct subgraph isomorphisms $F = \{f\}$ from g to G. $F(v)$ denotes the set of distinct vertices v' such that there exists an isomorphism f mapping $v \in V_g$ to $v' \in V_G$. The *minimum image based support*, or "support" in short, of g in G is $sup(g, G) = \min_{v \in V_g}\{|F(v)|\}$.

Example 4. Consider in Fig. 1deterministic graphs G and g, which model professional social networks with persons represented by vertices and collaboration by edges. Each vertex takes the profession of the person as its label, e.g., Bio indicates an biologist; edge labels are omitted for clarity. There are three subgraph isomorphisms between g and G, i.e., $f_1 = \{v_1, v_2\}$, $f_2 = \{v_3, v_2\}$ and $f_3 = \{v_3, v_4\}$. Consequently, $sup(g, G) = \min\{2, 2\} = 2$.

2.2 Uncertain Graph

An *uncertain graph* is a tuple $G^u = (G, P)$, where G is a deterministic backbone graph, and $P : E_G \rightarrow (0, 1]$ is a probability function that assigns each edge e with an existence probability, denoted by $P(e)$, $e \in E_G$. Upon the determination of each edge in G^u, a deterministic graph G^i is implied, i.e., *implied graph*, denoted by $G^u \Rightarrow G^i$. Hence, an uncertain graph G^u implies $2^{|E_G|}$ possible graphs in total, each of which is a structure that G^u may exist as.

We consider the model where independence among edges holds, which finds various applications [4], [6], [8]. The probability of G^u implying G^i, or *existence probability* of G^i, can be computed by including or excluding the edges:

$$P(G^u \Rightarrow G^i) = \prod_{e \in E_{G^i}} P(e) \prod_{e \in E_G \setminus E_{G^i}} (1 - P(e)).$$

While the classic notion of support becomes intriguing on uncertain graphs, we generalize the definition on uncertain graph databases [18], [19] to single uncertain graphs.

Definition 1 (Support in Uncertain Graph). *Support of a subgraph in a single uncertain graph is defined as the aggregation of supports of the subgraph over all implied graphs of the uncertain graph; formally,*

$$usup(g, G^u) = \sum_{G^i \in I(G^u)} P(G^u \Rightarrow G^i) \cdot \psi(g, G^i), \tag{1}$$

where $I(G^u)$ is the set of implied graphs of G^u, and ψ is a function to measure the contribution of support from an implied graph.

Intuitively, the contribution of support could be measured directly by the support in an implied graph,

$$\psi(g, G^i) = sup(g, G^i). \tag{2}$$

Equations (1) and (2) together actually measures the expected support of a subgraph g in G^u over all possible implied graphs, denoted by $esup(g, G^u)$.

It is also possible to measure the support by the probability that the support of g in G^u is no less than a given threshold. With this, ψ could be defined as

$$\psi(g, G^i) = [[sup(g, G^i) \geq \phi\ ]], \tag{3}$$

where $[[x]]$ denotes a binary indicator, which takes value 1 if the proposition x is evaluated to be true, and value 0 otherwise, ϕ is an integer between 0 and $sup(g, G)$. In nature, Equations (1) and (3) together defines the support of g in G^u under probabilistic semantics, denoted by $psup(g, G^u, \phi)$. For differentiation, we call the former as *expected support*, and the latter *probabilistic support*.

Example 5. Associating every edge of G in Fig. 1 with an existance probability, we construct an uncertain graph G^u, where probability models collaboration closeness between two persons. Hence, G is the backbone graph, and there are 8 deterministic graphs that can be implied by G^u. $P(G^u \Rightarrow G) = 0.4 \times 0.5 \times 0.6 = 0.12$, and $sup(g, G) = 2$. Therefore, (1) under expected semantics, $\psi(g, G) = 2$, and $P(G^u \Rightarrow G) \cdot \psi(g, G) = 0.24$. By accumulating the values from all implied graphs, the expected support $esup(g, G^u) = 1.12$; (2) under probabilistic semantics with $\phi = 2$, $\psi(g, G) = 1$, as $sup(g, G) \geq \phi$, and $P(G^u \Rightarrow G) \cdot \psi(g, G) = 0.12$. After taking all implied graphs into account, the probabilistic support $psup(g, G^u, \phi) = 0.24$.

Given an uncertain graphs G^u and a support threshold σ, a subgraph g is said to be *frequent* if $usup(g, G^u) \geq \sigma$. Support in uncertain graph is *anti-monotonic*.

Proposition 1 (Anti-monotonicity). *Consider two graphs $g \sqsubseteq g'$ over an uncertain graph G^u. (1) $esup(g, G^u) \geq esup(g', G^u)$; and (2) $psup(g, G^u) \geq psup(g', G^u)$.*

Proof. See Appendix B.1, available in the online supplemental material. □

Based on Proposition 1, we formulate two versions of FSM on single uncertain graphs under different semantics.

Problem 1. *Given an uncertain graph G^u, a positive integer ϕ and a support threshold σ, FSM on single uncertain graphs under probabilistic semantics finds all subgraphs g whose probabilistic support $psup$ is no less than the threshold, i.e., $\mathcal{G}_P = \{g \,|\, psup(g, G^u, \phi) \geq \sigma \wedge g \sqsubseteq G\}$.*

Problem 2. *Given an uncertain graph G^u, a support threshold σ, FSM on single uncertain graphs under expected semantics finds all subgraphs g whose expected support $esup$ is no less than the threshold, i.e., $\mathcal{G}_E = \{g \,|\, esup(g, G^u) \geq \sigma \wedge g \sqsubseteq G\}$.*

In the sequel, we present an enumeration-evaluation framework to solve both problems, and the evaluation of Problem 1 is investigated first, followed by extension to handle Problem 2. For ease of exposition, domain G^u and parameter ϕ are omitted onward when there is no ambiguity, e.g., "$psup(g, G^u, \phi)$" to "$psup(g)$".

3 Algorithm Framework

We present an *enumeration-evaluation* algorithm named fanta (frequent subgraph mining on uncertain graphs):

- *Enumeration*: enumerate all candidate subgraphs; and
- *Evaluation*: for each subgraph, compute its support, and decide whether to output as a result.

The enumeration phase is the same as that for FSM on a deterministic graph. Thus, any enumeration strategy leveraging the Apriori property can be used. The Apriori property states that supergraphs of an infrequent subgraph cannot be frequent. Specifically, all subgraphs of an uncertain graph can be organized in a rooted directed acyclic graph (DAG), where the nodes represent candidate subgraphs (with the root being $null$). An arc in the DAG from a pattern g' to g denotes that g' is a direct supergraph of g. We enumerate all possible subgraphs by starting from frequent single edges and attaching every time a new edge to those frequent subgraphs, so that subgraphs consisting of n edges can be found at level-n of the DAG. Only the children of a frequent subgraph will be enumerated. To avoid duplicate enumeration of a subgraph while retaining completeness, existing approach gSpan [21] imposes a lexicographic order among the subgraphs. We also employ this strategy for elegant enumeration of candidate subgraphs.

The evaluate phase determines whether an enumerated subgraph is frequent by comparing its support with the threshold. A naïve procedure is to generate all implied graphs, compute and aggregate the support contribution of the subgraph in every implied graph, and then derive the support and compare with the threshold. This can be rather time-consuming due to the large number of implied graphs and the high complexity of support computation, and hence, becomes unbearable to end-users. In practice, we seek every opportunity to return answers within reasonable time.

Since solutions to Problems 1 and 2 share the identical enumeration phase under the framework, in the following we present evaluation methods for them, separately.

4 Evaluation of Probabilistic Support

To evaluate a probabilistic support against support threshold, a straightforward approach is to compute the exact value of the probabilistic support, and compare it with the threshold. Nevertheless, computing probabilistic support directly by Equations (1) and (3) is excessively complex; that is, for each candidate subgraph, we need to calculate its support in an exponential number of deterministic graphs, where tedious subgraph support evaluation is frequently involved. In fact, we will shortly see that it is #P-complete to aggregate the probability from all frequent implied graphs, and we are in pursuit of an efficient evaluation method.

4.1 Evaluation Algorithm

We first look at the hardness of the problem.

Theorem 1. *It is #P-complete to compute* $psup(g)$.

Proof. See Appendix B.3, available in the online supplemental material. □

The proof of Theorem 1 implies that computing $psup(g)$ is #P-complete, even if the support of g in each implied graph is given, and the additional high complexity is mainly brought by the involvement of probabilities. In other words, computing $psup(g)$ is much harder than computing the support of g over a deterministic graph. To deal with the extra hardness (compared with FSM on deterministic graphs), we propose an approximate evaluation algorithm to avoid exact support computation to aggregate the probability of g.

As an approximation algorithm, it is desirable that a subgraph will be returned if it is frequent (true positive); to achieve this, we have to compromise with infrequent subgraphs in the result set (false positive). To this end, we interpret the output of an evaluation as a closed interval $[\underline{psup}(g), \overline{psup}(g)]$ that approximately contains the true value of $psup(g)$, and carefully handle the following cases when evaluating subgraph g against support threshold σ and the absolute error tolerance ϵ:

- *Case 1*: If $\overline{psup}(g) < \sigma$, do not output g, since it is certain that $psup(g) < \sigma$;
- *Case 2*: If $\overline{psup}(g) \geq \sigma$ and $\underline{psup}(g) \geq \sigma - \epsilon$, output g, as it is certain that $psup(g) \geq 1 - \epsilon$, and it is probable that $psup(g) \geq \sigma$; and
- *Case 3*: If $\overline{psup}(g) \geq \sigma$ or $\underline{psup}(g) < \sigma - \epsilon$, it cannot be determined whether or not to output g, because we cannot decide whether $psup(g) \geq \sigma$ or not.

By intuition, Case 3 is not desirable, as we cannot make a decision on g. Nevertheless, we observe that if the width of interval $[\underline{psup}(g), \overline{psup}(g)]$ is within ϵ, Case 3 will not happen. Thus, by enforcing the interval width to be at most ϵ, it is sufficient to approximate $psup(g)$ by the interval $[\underline{psup}(g), \overline{psup}(g)]$, where only Cases 1 and 2 happen. This is crucial to the algorithm design, and we rely on this to determine whether to include g as a result.

Our approximation algorithm is based on Monte-Carlo method. The requirement that the interval should be within ϵ recalls a class of randomized algorithms - *randomized approximation scheme* - that can provide accuracy guarantee [22]. Given real numbers $\varepsilon, \delta \in [0, 1]$, we can use the value $\hat{p}$ produced by a randomized approximation scheme to estimate p, if $P(|\hat{p} - p| < \varepsilon) \geq 1 - \delta$, where $1 - \delta$ is a confidence degree. Thus, to approximate probabilistic support under the constraints of δ and ε, we resort to Hoeffding's inequality stated in Lemma 1 in Appendix A, available in the online supplemental material.

Lemma 1 advises that the average mean of N sample observations provides an approximation of p with accuracy guarantee, and the sample size required for satisfying confidence degree $1 - \delta$ and $\varepsilon' = \varepsilon/2$ is

$$N \geq \frac{\ln(2/\delta)}{2\varepsilon'^2} = \frac{2\ln(2/\delta)}{\varepsilon^2}.$$

Lemma 1 can be directly applied to compute probabilistic support by sampling identically and independently from the implied graphs, and the mean of $\psi(g, G^i)$ (Equation (3)) over the sample graphs just approximates the probabilistic support, with the absolute error tolerance $\epsilon = \varepsilon$. Note that "sample graph" differs from "implied graph" in that two sample graphs may correspond to the same implied graph.

When drawing sample graphs from G^u, instead of sampling over all edges, we only consider the *embedding edges* of

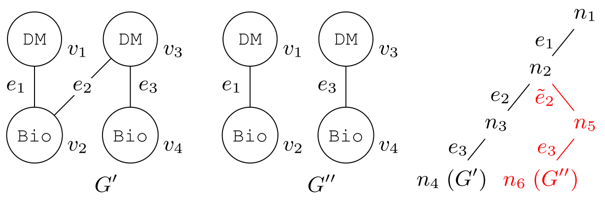


Fig. 2. Example of computation sharing.

subgraph g in the backbone graph G, i.e., $\mathcal{E}_m = \{e \,|\, e \in F(g)\}$, where F is the set of embeddings from g to G. This shrinks the sample space but does not affect the correctness of the result, as stated in Theorem 2. From now on, we still refer it as a sample graph of G^u, although it contains only (partial) embedding edges.

Theorem 2. *Sampling over embedding edges correctly produces the approximation result with the constraints of real numbers* $\varepsilon, \delta \in [0, 1]$.

Proof. See Appendix B.4, available in the online supplemental material. □

Algorithm 1 encapsulates the procedures for evaluating a subgraph. It takes as input a subgraph g, an uncertain graph G^u, a positive integer ϕ, an error tolerance ε and a real number δ; and it outputs a Boolean value that indicates whether g is frequent. In particular, we first find all embeddings of subgraph g in backbone graph G (Line 4.1), which is readily achieved through *grow-store*, based on the embeddings of the direct subgraph g'. Then, N sample graphs are drawn (Lines 2 – 3). We apply the Monte-Carlo simulations to derive the probabilistic support (Line 4), and evaluate it via EvaluateSupport (Lines 5 – 8).

Algorithm 1. EvaluateProbabilistic $(g, G^u, \sigma, \phi, \varepsilon, \delta)$

```
Input: g, the subgraph; G^u, the uncertain graph; σ, the
       support threshold; φ, a positive integer; ε, the error
       tolerance; δ, the confidence degree.
Output: Boolean value - true if g is a frequent subgraph;
        false, otherwise.
1 $F(g) \leftarrow$find all embeddings of $g$ in $G$ based on $F(g')$;
2 $N \leftarrow \frac{2\ln(2/\delta)}{\varepsilon^2}$;
3 $\Omega \leftarrow$ randomly draw $N$ implied graphs of $G^u$;
4 $p \leftarrow$ MonteCarloSimulate $(g, \Omega)$;
5 switch EvaluateSupport $(p)$ do
6   case 1 do return false;
7   case 2 do return true;
8   case 3 do do nothing;                /* invalid */
  Function MonteCarloSimulate $(g, \Omega)$
1   $X \leftarrow 0$;
2   foreach sample graph $G_i \in \Omega$ do
3     $sup \leftarrow$ ComputeSup$(g, G_i)$;
4     if $sup \geq \phi$ then$X \leftarrow X + 1$;
5   return $X/N$
```

In general, Function MonteCarloSimulate conducts the Monte-Carlo simulations to derive the value of p. Specifically, it uses a counter X to log the observations. On each sample graph, the support of g is calculated. The counter X is incremented by 1 if the support is no less than ϕ. After screening all samples, the empirical mean $\frac{X}{N}$ is returned, which provides an unbiased estimation of the real p.

Afterwards, procedure EvaluateSupport works as follows. It admits the approximated value p of $psup(g)$, and outputs either Cases 1 or 2 (Case 3 should not be reached). Particularly, it generates an interval $[\underline{psup}(g), \overline{psup}(g)]$, by which the probabilistic support is bounded, where $\underline{psup}(g) = \tilde{p} - \frac{\varepsilon}{2}$ and $\overline{psup}(g) = \tilde{p} + \frac{\varepsilon}{2}$. Upon the interval, we make decision against the threshold σ.*Correctness and Complexity.* The correctness of Algorithm 1 is guaranteed by Lemma 1, which ensures that any output subgraph meets the specified constraints.

We proceed to complexity analysis. The complexity of discovering embeddings of g from embeddings of g' is $O(|F(g')||V_G|)$. Given the discovered embeddings $F(g)$, we compute the support of g in G^i in $O(|F(g)||V_g|)$. As the number of samples is bounded by $O((\frac{1}{\varepsilon})^2 \ln \frac{1}{\delta})$, the overall complexity is hence $O(|F(g')||V_G| + \frac{1}{\varepsilon^2} \ln(\frac{1}{\delta})|F(g')||V_G||V_g|)$. Although $|F(g)|$ can be exponential theoretically, it is not generally in practice and thus Algorithm 1 is efficient to run.

4.2 Computation Sharing

A straightforward way to implement the Monte-Carlo based support evaluation is to compute subgraph support in each sample graph, and aggregate the number of observations. This can be time-consuming, and we go in quest of speedup.

Example 6. Consider uncertain graph G^u and subgraph g in Fig. 1. Fig. 2 presents two sample graphs G', G'' $(G' \sqsupseteq G'')$ of uncertain graph G^u. There are 3 embeddings of g in G', $F_{G'}(g) = \{f_1, f_2, f_3\}$, where $f_1 = \{v_1 - v_2\}, f_2 = \{v_3 - v_2\}$ and $f_3 = \{v_3 - v_4\}$, and $F_{G''}(g) = \{f_1, f_3\}$. After determining the embedding set of G', it is easy to derive the embeddings in G'' by removing those involving e_2. Besides, recall the definition of MI based support. When deriving $sup(p, G')$, we need to align the embedding vertices in $F_{G'}(g)$ to the corresponding vertex in g, e.g., $\{v_1, v_3\}$ for u_1. This temporary data structure can be leveraged to compute $sup(p, G'')$ by erasing the vertices of f_2.

Evidenced by the example, given two sample graphs G' and G'', we discover that two types of computational costs can be shared between them: (1) testing whether an embedding exists; and (2) computing support of subgraphs.

Next, we generalize the idea to multiple sample graphs where containment does not necessarily hold. Since Types (1) cost is directly related to the differences of single edges between two graphs, we examine it first and then Type (2).

4.2.1 *Sharing Type (1) Cost*

While it is complex to model the costs accurately, we observe that Type (1) cost is roughly proportional to the number of different edges between two graphs. Assuming that each different edge, which either exists in G_i but not G_j or vice versa, incurs a *unit cost* of Type (1), we put forward the following cost function to approximate the cost of incremental computation from G_i to G_j.

Given a global order of the embedding edges, we represent a sample graph G_i by a bit array $B_i = b_1 b_2 \cdots b_k \cdots b_{|\mathcal{E}_m|}$, where $b_k = [[e_k \in E_{G_i}]], k \in [1, |\mathcal{E}_m|]$. For instance, G' in Fig. 2 can be represented as $B' = 111$, and G'' as $B'' = 101$, respectively.

Definition 2 (Cost between Graphs). *The incremental computational cost from graph* G_i *to* G_j *is defined to be*

$$cost(G_i, G_j) = \min\{\|B_i \oplus B_j\|_1, \|B_j\|_1\},$$

where $\oplus$ is the exclusive OR operation between the bit arrays, and $\|\cdot\|_1$ is the ℓ_1-norm.

As a special case, the cost of processing the first graph G_1 equals $\|B_1\|_1$. By Definition 2, we can derive an order such that the sample graphs are examined in sequence, where we process G_{i+1} by the way with the smaller cost, either on the basis of G_i, $i \in [1, N-1]$ or from scratch. To optimize the overall cost, we formulate the following problem.

Problem 3. *Given a collection of sample graphs Ω, and use Definition 2 as the cost function, find a* linear order *of the graphs in Ω such that the aggregated cost between two consecutive graphs is minimized.*

Theorem 3. *It is NP-hard to solve Problem 3.*

Proof. See Appendix B.5, available in the online supplemental material. □

In practice, we propose to reduce the cost by dynamically constructing a computation sharing tree with heuristics. Regarding B_i and B_j as two numbers in binary, adapted from Definition 2, we approximate the cost from G_i to G_j by the minimum of (1) the difference from B_i to B_j, and (2) the absolute value of B_j.

Hence, the larger the *longest common prefix* that B_i and B_j have, the smaller the cost between G_i and G_j will be. This simplified version eases the construction and optimization of the sharing tree.

Theorem 4. *The approximation error is in $O(1)$ on average.*

Proof. See Appendix B.6, available in the online supplemental material. □

The computation sharing tree $\mathcal{T}$ is a binary search tree [23] with all sample graphs as leaf nodes. $\mathcal{T}$ has $|\mathcal{E}_m|$ levels, the sample graphs are at level $|\mathcal{E}_m|$, and the branching at level i is based on inclusion or exclusion of edge e_i in the sample graphs. In other words, the root is at level 0, and the left child n_l of the root leads to all sample graphs having e_1, while the right child n_r comprises the remaining graphs that do not have e_1. This branching process carries on from e_1 till the last in $\mathcal{E}_m$, and conceptually, there are $2^{|\mathcal{E}_m|}$ leaf nodes. Thus, we can find a position for every sample graph at the level of leaf nodes. On the right of Fig. 2 shows a partially constructed tree for G' and G''.

In our implementation, we do not generate the whole tree but only materialize a branch if there is a sample graph as its leaf. The dynamic construction procedure is formulated in Algorithm 2, where Type (1) sharing is achieved as a "side product". Specifically, given the set of sample graphs, we construct the sharing tree iteratively for each sample graph. We first create a new node as the root (Line 1). Then, for each sample graph, we materialize the branch for it (Lines 2-13). Specifically for G_i, it starts from the root (Line 3), and then traverses downward the tree until leaf level (Lines 4-13). If $G_i.b_i = 1$, edge e_i exists in G_i, and hence, the left child of the current node at level-$(i-1)$ would be visited next; otherwise, the right child. During the traversal, if it encounters an empty branch, the intermediate result would be reserved for potential sharing in future. Note that if the right child is empty, the embeddings containing edge e_i, which is maintained in an inverted index $\mathcal{I}$, are removed from the embedding set F (Lines 11-12).

Algorithm 2. ConstructShareTree(Ω)

```
    Input: $\Omega$, a set of sample graphs; $\mathcal{I}$, an inverted index of
           embeddings.
    Output: $\mathcal{T}$, a computation sharing tree.
 1  $\mathcal{T}$.root ← create a new node;
 2  foreach sample graph $G_i \in \Omega$ do
 3    $n \leftarrow \mathcal{T}$.root, $F \leftarrow F(G)$;
 4    for $i = 1$ to $|\mathcal{E}_m|$ do
 5      if $G_i.b_i = 1$ then
 6        $n \leftarrow$ the left child of $n$;
 7        if $n$ is null then
 8          create a new node $n$ such that $n.F \leftarrow F$;
 9      else
10        $n \leftarrow$ the right child of $n$;
11        if $n$ is null then
12          create a new node $n$ such that
            $n.F \leftarrow F \setminus \mathcal{I}(e_i)$;
13      $F \leftarrow n.F$;
14  return $\mathcal{T}$
```

Example 7. Further to Example 6, on the right of Fig. 2 is a partial tree for G' and G'', where n_1 is the root, and n_4 (resp. n_6) is a leaf node accommodating G' (resp. G''). After processing G', we have materialized a branch ($n_1 - n_2 - n_3 - n_4$). The result on node n_2 can be then shared when constructing the branch for G'' ($n_1 - n_2 - n_5 - n_6$).

Lastly, we discuss another enhancement injected into the tree construction. Recall that the optimization target is to maximize the longest common prefix between two bit arrays of graphs. Intuitively, the more skewed existence probability (closer to 0 or 1) an edge has, the more sample graphs may include (or exclude) the edge, and hence, the longer common prefix the graphs may share, and the more computation can be reused. Thus, in order to expect the maximum sum of the lengths of longest common prefix between sample graphs, we enforce an order of the embedding edges from large to small, in terms of the absolute difference between its existence probability and 0.5. In other words, the embedding edges are sorted as $e_1, e_2, \dots, e_{|\mathcal{E}_m|}$ such that $|P(e_1) - 0.5| \geq |P(e_2) - 0.5| \geq \cdots \geq |P(e_{|\mathcal{E}_m|}) - 0.5|$.

4.2.2 Sharing Type (1) Cost

After constructing the sharing tree, we calculate the support of a subgraph g on the sample graphs in a serial fashion. Given sample graphs G' and G'', we first describe how to compute $sup(g, G')$. For each embedding contained by G', we parallelize the *embedding vertices* to the vertices of g. Note that since we have already obtained the isomorphic mappings, this operation is much easier and faster than the general graph alignment. In specific, for a vertex v_g, we employ a map to keep a record - each embedding vertex in G' as a key, and the number of occurrences in all embeddings as its value. When we encounter an embedding

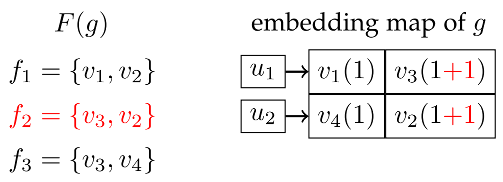


Fig. 3. Example of support computation.

having u as the embedding vertex for v_g, we increment the value of u in the map for v_g by 1. After counting all vertices in all embeddings, we derive $sup(g, G')$ by retrieving the size of the smallest map under every vertex of g.

Example 8. Fig. 3 is an example of embedding map of g in Fig. 1 based on $F(g)$. The map contains two keys, each corresponding to a vertex of g. Each key contains two entries; take u_1 as an example, where u_1 is mapped to two data vertices, v_1 and v_3, respectively. The number followed in the bracket showing the occurrences of the corresponding data vertex. Based on $\{f_1, f_3\}$, $sup(g)$ is obtained by determining the minimum size of the posting lists. As the sizes of posting lists of u_1 and u_2 are 2 and 1, respectively, $sup(g) = 2$. When a new embedding f_2 (in red) is found, we adjust the embedding map, and hence, $sup(g)$ becomes $\min\{2, 2\} = 2$.

Afterwards, to reuse the intermediate results for G'', instead of building the map from scratch, we eliminate the embeddings not contained by G'', decrementing the corresponding values in the map, and then append the embeddings not covered previously by G', incrementing the corresponding values. Then, we can derive $sup(g, G'')$.

Model the cost of a unit increment/decrement to be fixed as c, and that of retrieving the size of the smallest map to be c'. Given a collection of sample graphs G_i each containing a set of embeddings $F_i \subseteq F$, where $i \in [1, N]$, the cost of support computation from scratch is $|F_i| \cdot c + c'$. Define $m_i^j \triangleq |(F_i \cup F_j) \setminus (F_i \cap F_j)|$. The cost of support computation from G_i (resp. G_j) to G_j (resp. G_i) is $m_i^j \cdot c + c'$ (resp. $m_j^i \cdot c + c'$), and $m_i^j = m_j^i$. If $|F_j| < m_i^j$, it is less costly to compute for G_j from scratch, instead of from G_i.

Intuitively, to maximize the computation sharing, we wish for a computation sequence of the sample graphs. Similar to Problem 3, we can formulate an optimization problem. It is conjectured that the problem is also NP-hard, and hence, efficient implementation is sought. In particular, we adopt a heuristic - from left to right according to graph positions at the level of leaf nodes. The intuition is that two consecutive leaf nodes like G' and G'' in Fig. 2 are usually similar, thus incurring small Type (1) cost.

Compiling things together, we present Algorithm 3. To integrate it into the baseline algorithm, we replace Line 4 of Algorithm 1 with "$p \leftarrow$ ShareCompSimulate(g, Ω)".

5 Evaluation of Expected Support

Although expected support is defined differently from probabilistic support, we find that the aforementioned techniques in Section 4 could be extended to support evaluation under expected semantics. Nevertheless, due to the specialty of expected semantics, the sample size required becomes unfixed and usually much larger. In the sequel, we first show the connection between the two definitions, and then, present an evaluation method with a checkpoint mechanism for candidate pruning and validation.

Algorithm 3. ShareCompSimulate(g, Ω)

```
Input: $g$, a subgraph; $\Omega$, a set of sample graphs.
Output: $p$, the approximate value of $psup$.
$\mathcal{I} \leftarrow$ construct the inverted index using $F(g)$
$\mathcal{T} \leftarrow$ ConstructShareTree$(\Omega, \mathcal{I})$
$X \leftarrow 0$
reorder $\Omega$ as per leaf nodes from left to right in $\mathcal{T}$
$sup \leftarrow$ ComputeSup$g, G_1$1st in $\Omega$
if $sup \geq \phi$ then $X \leftarrow X + 1$
foreach sample graph $G_i (i \in [2, |\Omega|]) \in \Omega$ do
    $m \leftarrow |(F_i \cup F_j) \setminus (F_i \cap F_j)|$
    if $m < |F_i|$ then
        $sup \leftarrow$ ShareComputeSup$(g, G_{i-1}, G_i)$
    else $sup \leftarrow$ ComputeSup$(g, G_i)$
    if $sup \geq \sigma$ then $X \leftarrow X + 1$
return $X/N$
```

5.1 Evaluation Algorithm

We first present a reformulation of Equations (1) and (2) to show the connection between expected and probabilistic support. Let $P(sup(g) = j)$ denote the aggregated probability that the support of g in an implied graph equals j,

$$P(sup(g) = j) = \sum_{G^i \in I(G^u)} P(G \Rightarrow G^i)[[\text{sup}(g, G^i) = j]].$$

It is not difficult to rewrite Equations (2) and (1) as

$$esup(g) = \sum_{j=1}^{M_s} P(sup(g) = j) \cdot j, \tag{4}$$

where $M_s = sup(g, G)$ is the maximum support of g in all implied graphs of G^u, namely, $sup(g, G)$. Further, denote as $P_j(g)$ the aggregated probability that the support of g in an implied graph is no less than j, i.e.,

$$P_j(g) = \sum_{G^i \in I(G^u)} P(G^u \Rightarrow G^i)[[\text{sup}(g, G^i) \geq j]].$$

Hence, $P(sup(g) = j) = P_j(g) - P_{j+1}(g)$.

Proposition 2 (Reformulation). $esup(g) = \sum_{j=1}^{M_s} P_j(g)$.

Proof. See Appendix B.2, available in the online supplemental material. □

Theorem 5. *It is #P-hard to compute $esup(g)$.*

Proof. See Appendix B.7, available in the online supplemental material. □

Due to the complexity of computing $esup(g)$, an approximation algorithm is proposed with error tolerance ε. Similarly, we interpret the output as a closed interval $[\underline{esup}(g), \overline{esup}(g)]$ that approximately contains the true value of $esup(g)$. As to the relations between the interval and support threshold, three cases are carefully handled (cf. Section 4.1). Note $\epsilon = \varepsilon\sigma$, ε is a real number in $[0, 1]$ and σ is the threshold. Moreover, as undecidable case is not desired,

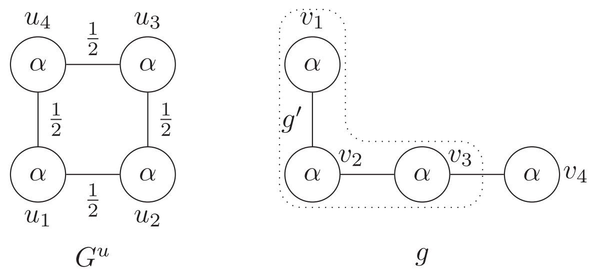


Fig. 4. Counter example of intuitive bound.

we enforce the interval width to be within $\varepsilon\sigma$. To derive the required sample size, we resort to the general case of Hoeffding inequality in Lemma 2 in Appendix A, available in the online supplemental material.

Hence, we can derive the required sample size as follows. Given a candidate subgraph g and an uncertain graph G^u, we draw N samples $G_1, G_2, \dots, G_N$ independently from G^u, and the corresponding support of g is denoted by $s_1, s_2, \dots, s_N$, respectively, each of which is bounded by $[0, M_s]$. By incorporating Lemma 2, we have

$$P\left(|\frac{1}{N}\sum_{i=1}^{N} s_i - s| \geq \varepsilon'\right) \leq 2\exp\left\{-\frac{2n\varepsilon'^2}{M_s^2}\right\}.$$

Thus, $N = \frac{2M_s^2 \ln(2/\delta)}{\varepsilon^2\sigma^2}$ samples are required at least. We describe the baseline method for evaluating expected support:

- *Sampling*: draw required number of sample graphs;
- *Simulation*: conduct Monte-Carlo simulations by calling MonteCarloSimulate ; and
- *Decision*: determine whether the candidate subgraph is a result by calling EvaluateSupport.

In particular, procedures MonteCarloSimulate and EvaluateSupport work similarly as in Algorithm 1. The difference lies in that the output of MonteCarloSimulate here is an array $\mathcal{P}$ of probabilities, where $\mathcal{P}[i]$ represents the observed value of $P_i(g)$. As the sample size required by the expected support is much larger than that by probabilistic support, we go inquest of further optimizations.

5.2 Pruning at Checkpoints

We investigate the option of early determination of whether the candidate subgraph is frequent or not.

5.2.1 Checkpointing Mechanism

While $N = \frac{2{M_s}^2\ln(\frac{2}{\delta})}{\varepsilon^2\sigma^2}$ samples are required to guarantee that undecidable Case 3 would not happen, we could make decision for a subgraph g earlier before the confirmation on interval width, if Cases 1 or 2 has already been satisfied. In this case, we can stop simulation earlier, and hence, save computation. A subsequent extreme of using this idea is to check the conditions at every sample, where the loss may outweigh the gain. To achieve the best of both worlds, we incorporate a periodical checkpointing mechanism such that early termination can be achieved at certain checkpoints.

Checkpoints are given sample sizes when periodical check is conducted. Suppose we check r times, when the sample size reaches $c_1, \dots, c_k, \dots, c_r$, respectively, where $0 = c_0 < c_1 < \cdots < c_k < \cdots < c_r = N$. For each checkpoint, we conduct incremental computation over $c_{k+1} - c_k$ graphs. In specific, as shown in Algorithm 4, Lines 4 – 6 retrieve the sample graphs required from Ω. Then, Line 7 computes with Ω' the current observation probabilities $P_j(g)$'s. Eventually, Line 8 determines whether to output the subgraph by EvaluateSup. Note that Case 3 here is not always invalid when it has not reached the terminal checkpoint.

Algorithm 4. CheckpointEvalExpected(Ω)

```
   Input: $\Omega$, the sample graphs.
   Output: boolean - **true**, if $g$ is a frequent subgraph;
           **false**, otherwise.
1  $N = \frac{2M_s^2\ln(2/\delta)}{\varepsilon^2\sigma^2}$;
2  $\Omega \leftarrow$ randomly draw $N$ implied graphs of $G^u$;
3  for $k = 0$ to $r - 1$ do
4      $c \leftarrow c_{k+1} - c_k$;
5      $\Omega' \leftarrow$ the first $c$ sample graphs in $\Omega$;
6      $\Omega \leftarrow \Omega \setminus \Omega'$;
7      $\mathcal{P} \leftarrow$ MonteCarloSimulate($\mathcal{P}, \Omega'$);
8      switch EvaluateSupport ($\mathcal{P}$) do
9          case 1 do return false;
10         case 2 do return true;
11         case 3 do do nothing;
```

Particularly, we select M_s checkpoints based on the discussion in Appendix C.3, available in the online supplemental material, where $c_k = \frac{2M_s^2\ln(2/\delta)}{(M_s+1-k)^2\varepsilon^2\sigma^2}$, $k = 1, \dots, M_s$. This selection ensures that at checkpoint c_k, the overall error of $\sum_{j=1}^{k} P_j(g)$ would not exceed $\varepsilon\sigma$.

Theorem 6. *The choice of the M_s checkpoints is optimal.*

Proof. See Appendix B.8, available in the online supplemental material. □

5.2.2 Structure-Based Bounds

In the basic implementation of EvaluateSupport, an interval is obtained in terms of absolute error. However, absolute error can be quite large, especially at early checkpoints. Reversely, if there is a tight bound to assist our decision, the performance can be improved. In nature, we attempt to find tighter bounds by considering structural properties, rather than obtain bounds directly in terms of absolute error.

An upper bound $\mathcal{U}(g)$ of $esup(g)$ is used to prune a subgraph; that is, if $\mathcal{U}(g) < \sigma$, g is determined not to be frequent. By Apriori property, $P_j(g)$ is bounded by $P_j(g')$, where g is a direct supergraph of g'. While this inequality is straightforward, can we enhance its pruning power? An immediate idea is to multiply $P_j(g')$ with $P_j(e)$, where e is the additional edge in g. However, this is *not* always correct in general, and we show an example for illustration.

Example 9. Consider uncertain graph G^u, subgraphs g and g' in Fig. 4. Note that both for subgraphs g' and g, among all implied graphs, only the backbone graph G can provide support 4. Hence, $P_4(g) = P_4(g') = \frac{1}{16}$. As $P_4(e) = \frac{7}{16}$, where $e = g \setminus g'$, $P_4(g) > P_4(g') \cdot P_4(e)$.

While the result is somewhat counter-intuitive, we argue that this is due to the special constraints of vertex-based support definition on single (uncertain) graphs. Nonetheless, thanks to Theorem 7, we can still leverage the pruning rule in many cases in practice.

Theorem 7. *Consider two graphs g and g', where g is a direct supergraph of g' with an extra edge e, and $e \not\sqsubseteq g'$.*

$$P_j(g) \leq P_j(g') \cdot P_j(e).$$

Proof. See Appendix B.9, available in the online supplemental material. □

To apply the pruning rule in Theorem 7, we define $I = [[e \not\sqsubseteq g']]$. That is, when e is distinct from all edges in g', $I = 1$ advises that we can leverage the tighter bound; otherwise, the Apriori-based bound is employed.

Formally, each probability $P_j(g)$ is bounded by

$$u_j(g) = \min\{\overline{P_j}(g),\ u_j(g') \cdot u_j(e)^I\},$$

where $\overline{P_j}(g) = \hat{P}_j(g) + \frac{\varepsilon\sigma}{2r}$, r indicates the current checkpoint, and $\frac{\varepsilon\sigma}{2r}$ is the current error tolerance to ensure the confidence. $u_j(g')$ and $u_j(g)$ denote the upper bound of $P_j(g')$ and $P_j(e)$, respectively. Since both $u_j(g')$ and $u_j(e)$ are necessary, in our implementation, we find all the frequent edges prior to the pattern growth. Thus, $u_j(e)$ could be calculated and stored globally when e is evaluated. We can also obtain $u_j(g')$ easily, as the support of g' has just been evaluated. Then, the upper bound of the expected support can be derived by summing up all the upper bounds of probabilities $P_j(g)$, which is $\mathcal{U}(g) = \sum_{j=1}^{M_s} u_j(g)$.

To incorporate upper bound $\mathcal{U}(g)$, we modify the evaluation procedure as Algorithm 5, and replace Line 8 of Algorithm 4 with "**switch** PruneEvalSupport $(\mathcal{P}, k)$".

Algorithm 5. PruneEvalSupport $(\mathcal{P}, k)$

```
Input: $\mathcal{P}$, an array of probabilities; $k$, the ordinal of the
       current checkpoint; $\mathcal{L}$ is the globally stored
       lower bound
Output: Case 1, 2 or 3.
1 $\mathcal{U} \leftarrow \mathcal{L}' \leftarrow I \leftarrow 0, \varepsilon' \leftarrow \frac{\varepsilon\sigma}{2k}$;
2 if $e \not\sqsubseteq g'$ then $I \leftarrow 1$;
3 for $j = 1$ to $M_s$ do
4     $u_j(g) \leftarrow \min\{\mathcal{P}[j] + \varepsilon', u_j(g') \cdot u_j(e)^I\}$;
5     $\mathcal{U} \leftarrow \mathcal{U} + u_j(g)$;
6 if $\mathcal{U} < \sigma$ then return 1;
7 for $j = 1$ to $M_s$ do $\mathcal{L} \leftarrow \mathcal{L} + l_j(g)$;
8 if $\mathcal{L} \geq \sigma - \sigma\varepsilon$ then return 2;
9 return 3;
```

Inside Algorithm 5, we also conceive a lower bound for validating subgraphs ($l_j(g)$ in Line 7). Inspired by Equation (1), instead of enumerating $2^{|\mathcal{E}_m|}$ graphs, we aggregate only the productions of support and probability from the unique graphs. Thus, $P_j(g)$ is also bounded by

$$l_j(g) = \max\{\underline{P_j}(g), \sum_{G^i \in \Omega^*} P(G^u \Rightarrow G^i)[[\sup(g, G^i) \geq j]]\},$$

where Ω^* is the set of unique graphs seen so far, which can be found easily from the sharing tree. The lower bound could be evaluated during MonteCarloSimulate lightly.

The correctness of the lower bound is immediate, since $\Omega^* \subseteq I(G^u)$. When incorporated with the checkpoint mechanism, the lower bound gets tighter with more sample graphs. The effect of the bounds is evaluated in Section 6.4, and empirical results indicate that the lower and upper bounds substantially help reduce the sample sizes.

6 Experiments

This section reports experiment results and analyses.

6.1 Experiment Setup

The proposed algorithms were implemented using C++ with STL support, running on a Linux machine with two Core Intel Xeon CPU 2.2Ghz and 16GB main memory. Particularly, four algorithms were implemented:

- fanta-PB: baseline for FSM under probabilistic semantics;
- fanta-P: fanta-PB equipped with the proposed computation sharing technique;
- fanta-EB: baseline for FSM under expected semantics; and
- fanta-E: fanta-EB enhanced by the proposed checkpoint-based pruning technique.

Experiments were carried out on three real-life networks:

- *PPI*[3]: PPI is the protein-protein interaction network, consisting of 4,631 vertices, representing the Clusters of Orthologous Groups (COG). The edges between COGs indicate the interactions, where the probability is provided by the STRING database. As edges exist between all COGs, forming a complete graph, we set a threshold such that if the existence probability of an edge is no larger than the threshold, the edge was removed. Finally, 10,747 edges were retained. The vertex labels were extracted from the information of COGs, denoting their functions.
- *CITE*[4]: CITE is a typical citation network of published articles, which consists of ~3k publications (vertices) and ~4k citations between them (edges). Each vertex has a single label representing an area under computer science, with 6 distinct labels in total. Each edge has a label (from 0 to 100) that measures the dissimilarity between the corresponding pair of publications. Hence, a smaller value indicates stronger similarity, and similarity is captured by edge probability in uncertain graphs, scaled to [0, 1].
- *DBLP*[5]: DBLP is a professional social network extracted from DBLP, including 100k distinct authors and 350k collaboration edges. The vertex label indicates the major direction that a person works on, and the edge probabilities express the strength of the collaboration between the incident authors. Particularly, the probabilities were derived from an exponential CDF of mean μ to times of collaboration; if two authors collaborated t times, the corresponding probability is $1 - e^{t/\mu}$ [15], [16], [17]. Following the convention, we considered $\mu = 5$ in our experiments.

3. http://string-db.org
4. http://linqs.cs.umd.edu/projects//projects/lbc/
5. http://www.informatik.uni-trier.de/~ley/db/

TABLE 1
Dataset Statistics

Dataset	$\|V\|$	$\|E\|$	$\|l_V\|$	$\|E\|/\|V\|$	avg $P(e)$
PPI	4,631	11,859	26	2.56	0.368
CITE	3,312	4,435	6	1.34	0.115
DBLP	100,000	349,684	10	3.50	0.52

We summarize the characteristics of the datasets in Table 1. For comparison, these uncertain graphs are of different structural characteristics. The density of graph is measured by $|E|/|V|$. DBLP is the largest network, which has the most vertices and the highest average existence probability, and is also the densest. The size of PPI is moderate, however, it has the most number of distinct vertex labels. Relatively, CITE is the smallest and sparsest.

6.2 Comparing Different Semantics

As fanta-P and fanta-E are developed under different semantics, it is of interest to investigate and cross-validate the results of fanta-P and fanta-E. Specifically, we recorded the results by fanta-P and fanta-E on PPI , with $\varepsilon = 0.1$ and $\delta = 0.1$. For fanta-P, we set $\phi \in \{20, 25, 30\}$ and varied σ from 0.25 to 1.0, whereas for fanta-E, we set $\sigma \in \{20, 25, 30\}$. Denote $\mathcal{G}_P$ (resp. $\mathcal{G}_E$) for the set of subgraphs returned by fanta-P (resp. fanta-E). We put forward the following two measures for appreciating $\mathcal{G}_P$ and $\mathcal{G}_E$:

$$deg_e = \frac{|\mathcal{G}_P \cap \mathcal{G}_E|}{|\mathcal{G}_E|} \quad \text{and} \quad deg_p = \frac{|\mathcal{G}_P \cap \mathcal{G}_E|}{|\mathcal{G}_P|},$$

where deg_e indicates the degree of subgraphs found by fanta-E can be discovered by fanta-P, and reversely, deg_p indicates the extent of subgraphs found by fanta-P to be observed by fanta-E.

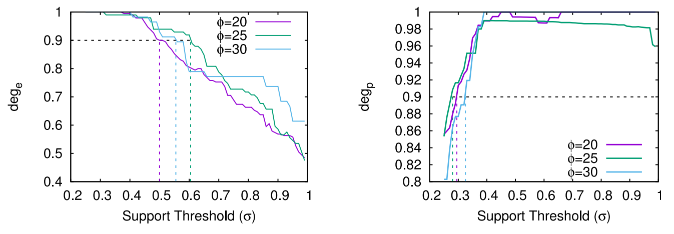
Fig. 5a and 5b illustrate the results of deg_e and deg_p, respectively, against the support threshold σ. Three lines are given, in accordance with $\phi = \{20, 25, 30\}$, each in reference with fanta-E of $\sigma = 20, 25, 30$, respectively. Fig. 5a shows that deg_e drops with the increase of σ, while Fig. 5b shows a growing trend. This is because higher support threshold reduces the number of subgraphs returned by fanta-P, and the result of fanta-P is harder to cover that of fanta-E, but easier to be identified by fanta-E.

Furthermore, to ensure deg_e no less than 0.9, the support threshold σ for fanta-P should not exceed around 0.500, 0.605 and 0.555, respectively, for $\phi = 20$, 25, 30, while σ for fanta-P should be no less than around 0.295, 0.280 and 0.325, accordingly. In short, the result difference could be diminished if the parameters are carefully selected.

6.3 Evaluating Approximation Quality

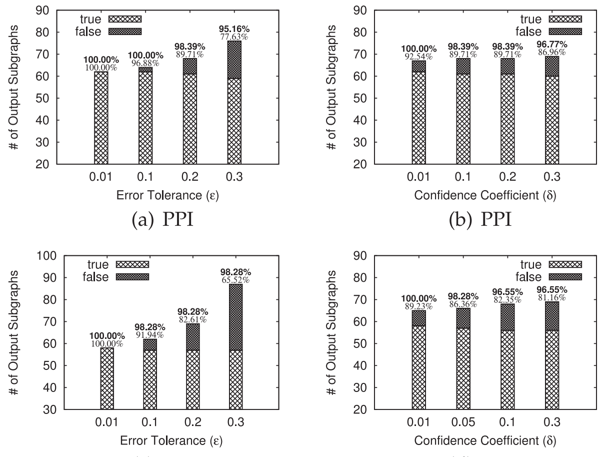
We evaluate the approximation quality of the algorithms fanta-PB and fanta-EB. Recall that the exact algorithm has to enumerate all possible implied graphs. Since it is infeasible to exactly find all the true frequent patterns on even slightly large graphs, we conducted the experiments on a small portion of PPI with 100 vertices, and regarded the subgraphs discovered under $\varepsilon = 0.01$ and $\delta = 0.01$ as the ground truth.

We investigate approximation quality with respect to parameters ε and δ, in terms of *precision* and *recall* - precision



(a) PPI (b) PPI

Fig. 5. Experiment results - semantic difference.



(a) PPI (b) PPI (c) PPI (d) PPI

Fig. 6. Experiment results - approximation quality.

is the percentage of true frequent subgraphs in the output, and recall is the percentage of output subgraphs in the true frequent subgraphs. The comparison results with respect to ε and δ are presented in Fig. 6a, 6b, 6c, 6d. The percentages above the each bar indicate the precision (in bold) and recall rates, respectively. In Fig. 6a and 6c, we varied ε from 0.01 to 0.3 and δ remained 0.1; while in Fig. 6b and 6c, we varied δ from 0.01 to 0.3 and ε remained 0.1.

It is revealed that both precision and recall decrease with the growth of ε and δ. We observe that the decreasing trend is more evident on precision against ε than that against δ, but neck and neck in recall. Moreover, the result of fanta-P is better approximated than that of fanta-E, which is demonstrated by the slow descent in precision and recall. We provide three possible reasons: (1) when ε becomes larger, more false frequent subgraphs are returned, hence reducing the precision; (2) when δ becomes larger, both false frequent and false infrequent subgraphs are enlarged, resulting in the decrease; and (3) as ε represents the relevant error tolerance in fanta-E, while in fanta-P it is the absolute error tolerance, the effect of ε is more evident on fanta-E than fanta-P. In conclusion, the proposed approximation algorithms are of high approximation quality.

6.4 Evaluating Proposed Techniques

This set of experiments evaluate the effectiveness of the proposed techniques under the approximation framework.

We first evaluate the sharing technique proposed in Section 4.2. Fig. 7a, 7b, 7c, 7d, 7e, 7f plot the elapsed time of fanta-P and fanta-PB on different datasets. We fixed parameters ε and δ both at 0.1, and varied ϕ from 15 to 20, σ from 0.1 to 0.9, respectively. The figures show that the execution

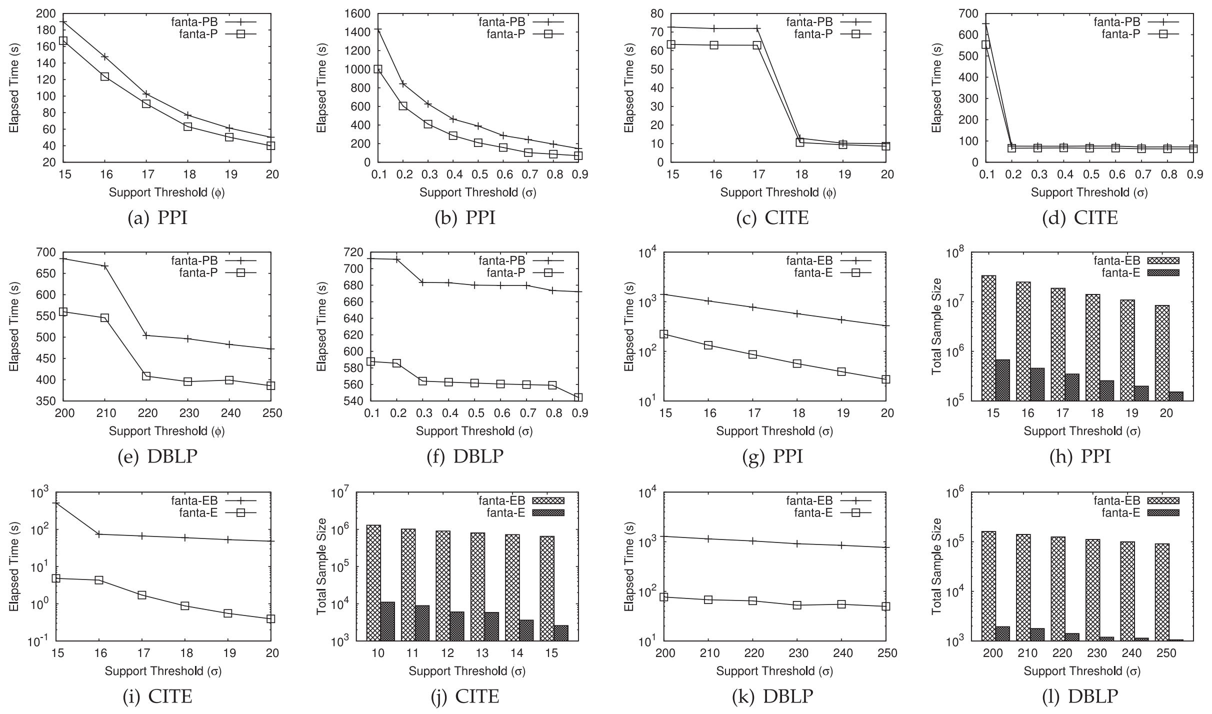


Fig. 7. Experiment results - proposed techniques.

time drops gradually. Moreover, by incorporating the computation sharing technique, fanta-P demonstrates superiority against fanta-PB, boosting the performance. Specifically, when it comes to CITE , minor advancement is shown after employing the sharing technique for varying σ in Fig. 7d; we also observe a steep jump when $\phi = 18$. These may be attributed to the special characteristic of CITE. On DBLP , we set higher support threshold ϕ towards the large volume of the network, where the performance gap between fanta-P and fanta-PB is more evident.

Next, we evaluate the pruning and validation technique proposed in Section 5.2, and present the results in Fig. 7g, 7h, 7i, 7j, 7k, 7l by fanta-E and fanta-EB. Besides running time, we record the sample size to show the effectiveness of the techniques. Specifically, on PPI , it is shown in Fig. 7g that the pruning and validation techniques remarkably improve the efficiency. In particular, when $\sigma = 15$, the upper bound reduces 71.4 percent of the samples, while the lower bound saves another 16.5 percent; when $\sigma = 20$, the upper and lower bounds reduce 78.3 and 11.1 percent of the samples. Thus, the lower bound based validation is more effective with small thresholds. As a consequence, fanta-E is an order of magnitude faster than fanta-EB. On CITE , about 10^6 samples are required by fanta-EB, which is reduced by 2 orders of magnitude by fanta-E. Through comparison with 10^7 (resp. 10^6) for fanta-EB (resp. fanta-E) on PPI, it indicates that the necessary sample graphs are much less on CITE, and thus, sharing does not enable a noteworthy outperformance. For Fig. 7e – 7l on DBLP , as the network is of larger volumn, we set higher support threshold σ to ensure the programs finish in reasonable time. The results further appreciate the techniques. Therefore, we safely come to the conclusion that fanta-P shows superiority over fanta-PB, and similarly, fanta-E over fanta-EB.

6.5 Evaluating Impact of Parameters

Apart from σ, parameters ε and δ also influence the algorithms. We measure the degree of influence by varying ε and δ from 0.01 to 0.3, respectively, where fanta-P versus fanta-PB, and fanta-E versus fanta-EB are evaluated, separately. Parameter settings are detailed in Table 2.

Fig. 8a, 8e and 8i demonstrate the influence of error tolerance ε on efficiency. The running time drops with the increase of ε, and then gradually levels off. On the other hand, the higher precision required, i.e., the lower error ε is to be tolerated, the more time should be sacrificed. The decreasing trend can also be witnessed in Fig. 8b, 8f and 8j, where the running time drops steadily. This illustrates that the influence of δ on the performance is moderate in contrast to ε.

For fanta-E and fanta-EB, the same experiments were conducted, producing Fig. 8c, 8g, 8k and 8d, 8h, 8l. While similar trend is observed, but fanta-EB is more sensitive to the variation of ε than fanta-E.

In summary, both fanta-P and fanta-E are less sensitive to the variation of ε and δ than the baselines fanta-PB and fanta-EB, respectively. Additionally, all the algorithms are influenced more by ε than δ. This is justifiable, as the time complexity of the algorithms is proportional to $\frac{1}{\varepsilon^2}$ and $\ln \frac{1}{\delta}$.

6.6 Evaluating Scalability

The scalability of fanta-P and fanta-E are evaluated in this set of experiments from two aspects - dataset size and density. Data size is measured by the number of vertices with

TABLE 2
Parameter Settings

Dataset	fanta-P (-PB)	fanta-E (-EB)
PPI	$\phi = 15, \sigma = 0.5$	$\sigma = 18$
CITE	$\phi = 10, \sigma = 0.5$	$\sigma = 35$
DBLP	$\phi = 200, \sigma = 0.5$	$\sigma = 200$

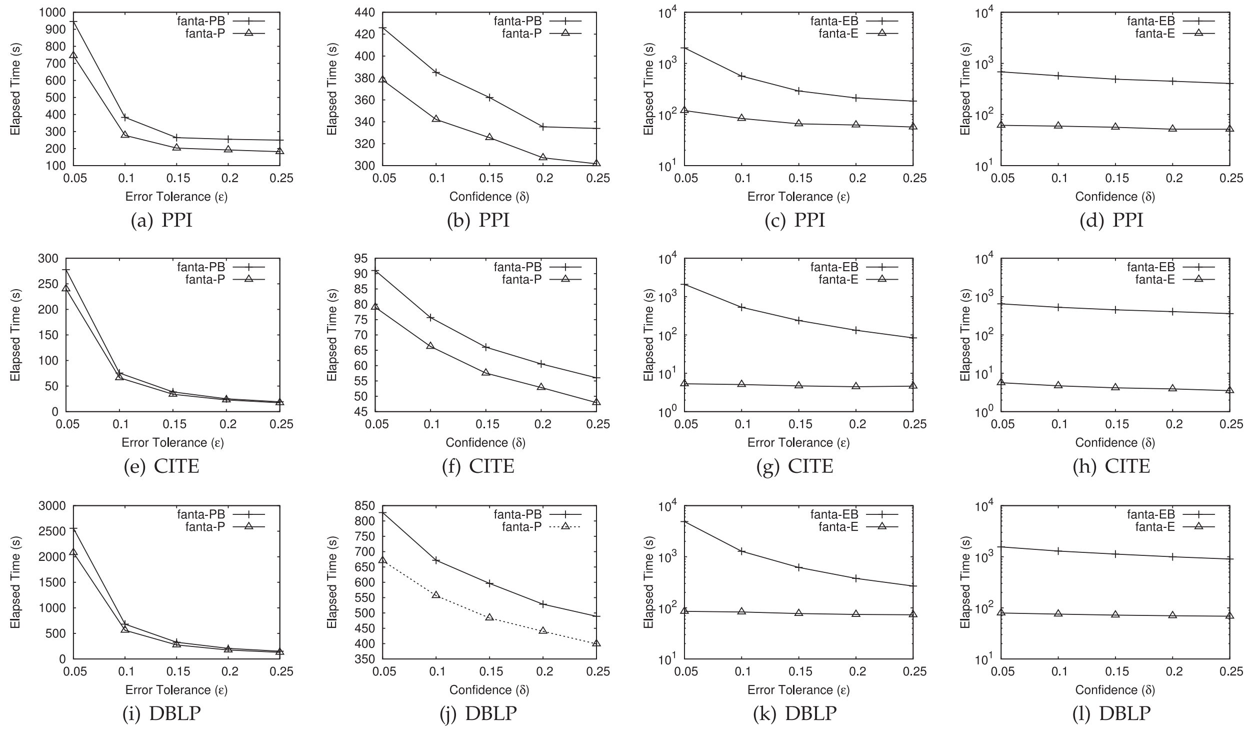


Fig. 8. Experiment results - impact of parameters.

the same density; for density, we varied the ratio of $\frac{|E|}{|V|}$ while fixing $|V|$. The graphs were synthesized from DBLP, where the corresponding subgraphs are generated through sampling. To reserve the property of original graph and ensure the connectivity of subgraphs, we adopt *Forest Fire* [24] for subgraph sampling.

First, fanta-P and fanta-E were evaluated on 5 samples from DBLP , respectively, 60, 80, 100, 120 and 140 percent of the default size, where $\phi = 200, \sigma = 0.5$ for fanta-P $\sigma = 200$ for fanta-E, and density equal 3.5. Fig. 9a shows linear growth by fanta-P, while fanta-E is less sensitive to the increase of dataset size. Note that with the increase, the number of subgraphs to be discovered mounts in an exponential rate. Thus, the results demonstrate good scalability by both algorithms, but more evident on fanta-E.

The experiment against graph density was run on a sample of DBLP with 2,000 vertices, varying the number of edges. The x-axis in Fig. 9b is density, i.e., $|E| = |V|$, $2|V|$, $3|V|$, $4|V|$, $5|V|$, respectively, and σ was fixed to 100 for fanta-E (or $\phi = 100, \sigma = 0.5$ for fanta-P). The figure reads an exponential increase in the elapsed time, where fanta-E grows much slower than fanta-P.

The scalability tests suggest that fanta-P and fanta-E can handle reasonably large and dense real-life graphs as those existing algorithms for deterministic graphs, while fanta-E scales better compared with fanta-P. This can be attributed to the proposed pruning and validation technique, which dramatically reduces the required sample size.

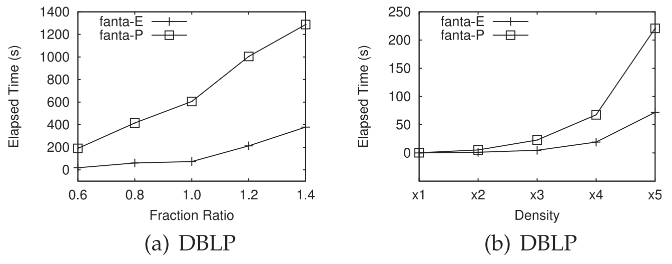


Fig. 9. Experiment results - scalability.

6.7 Case Study

In addition to performance evaluation, we conducted a case study on protein function prediction to validate the practicality of our proposed method, which is based on the method reported by [10]. The experimental details are described in Appendix C.5, available in the online supplemental material.

The prediction accuracy is shown in Fig. 10. The results suggest that the proposed methods can be applied to predict protein function, and they improve the prediction accuracy after taking into account uncertainties among protein interactions.

7 Related Work

Following discusses related work in three directions.

Mining Uncertain Graphs. All existing work on FSM on uncertain graphs is established on transaction settings; that is, there are multiple small/medium uncertain graphs in the database. Seeing various applications in bio-informatics and chem-informatics, Zou et al. proposed to mine frequent subgraphs from uncertain graph transactions under expected semantics [18], where a subgraph is frequent if its expected support is greater than the given threshold. MUSE algorithm was proposed to address the NP-hard problem, which is a combination of exact and approximation algorithms. In quest of better mining performance, the algorithm was later improved by leveraging edge and connectivity indices [25].

FSM under probabilistic semantics was also investigated [19], where a subgraph is frequent if its ϕ-frequent probability is greater the given threshold.

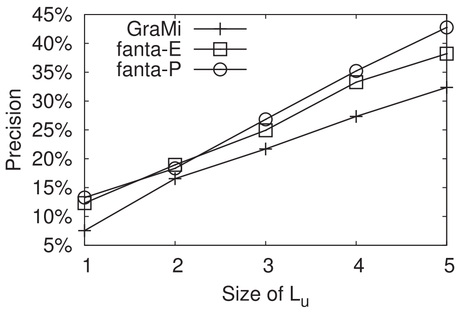


Fig. 10. Case study - precision of prediction.

Our research is inspired by the aforementioned work [18], [19], and especially the preliminaries set a good foundation of our work. While the basic definitions and preliminaries are adapted or extended in this paper, there exist great differences ascribable to the fact that we focus on single large uncertain graphs. Thus, our problem is more challenging due to not only the #P-hardness of support computation, but also the difficulty of elegant handling of vertex-based support and edge-based uncertainty.

Mining maximal cliques is of interests on uncertain graphs [26]. The concept of reliable subgraphs was uniquely established on uncertain graphs for subgraphs with high confidence [27], [28]. In addition, other mining problems include uncertain graph clustering [29], [30], core decomposition [31], representative instance extraction [32], etc.

Mining Deterministic Graphs. This line of research focuses on mining single deterministic graphs. Efforts were first dedicated to defining appropriate support measures, e.g., MI, HO and MIS. SIGRAM uses MIS, and follows a *grow-store* approach [33] - it needs to store the intermediate results for support evaluation. To avoid the computational complexity of MIS, HO [34] and MI [35] were proposed. The most recent work is attributed to GraMi [36], which formulates the single graph mining problem as a *constrained satisfaction problem*. A number of heuristic techniques were presented to enhance the baseline algorithm.

Among others, we are aware of several approximate algorithms, e.g., Grew [37] and gApprox [38]. For more graph pattern mining problems and algorithms, e.g., gSpan [21] and successors [39], readers may refer [9] for a recent survey.

Querying Uncertain Graphs. Querying uncertain graphs has received much attention. A natural problem to ask is on reachability [17] and shortest distances [40]. As to complex structures, sub(super)-graph search [41], [42], [43], matching and similarity search [44], [45] were investigated, where *filter-refine* algorithms were conceived to retrieve answers. Additionally, the concept of structural-context similarity was proposed to assist uncertain graph analysis [46]. To improve query efficiency and accuracy, a recursive stratified sampling was presented for query evaluation [47].

8 Conclusion

In this paper, we have investigated FSM on single uncertain graphs. Under an enumeration-evaluation framework, we propose an effective algorithm to achieve elegant mining performance. The #P-hard support computation is resolved by approximating the true value into an interval with accuracy guarantee. For probabilistic semantics, we develop computation-sharing heuristics to enhance the efficiency. For expected semantics, we prune unpromising and validate promising subgraphs, respectively, at checkpoints by utilizing structure-based upper and lower bounds. Extensive experiments indicate that the methods yield a promising solution for mining frequent subgraph on real-life graphs.

Acknowledgments

X. Zhao was supported by NSFC Nos. 61402498, 71690233, and 71331008; X. Lin by ARC DP150102728, DP170101628, DP180103096, and NSFC No. 61232006; and D. Guo by NSFC No. 61772544, and National Basic Research Program (973 Program) No.2014CB347800.

References

[1] E. Adar and C. Re, “Managing uncertainty in social networks,” *IEEE Data Eng. Bull.*, vol. 30, no. 2, pp. 15–22, Jul. 2007.
[2] P. Boldi, F. Bonchi, A. Gionis, and T. Tassa, “Injecting uncertainty in graphs for identity obfuscation,” *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1376–1387, 2012.
[3] S. Biswas and R. Morris, “ExOr: Opportunistic multi-hop routing for wireless networks,” in *Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, 2005, pp. 133–144.
[4] 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 *Proc. 26th IEEE Int. Conf. Comput. Commun.*, 2007, pp. 1721–1729.
[5] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, “Approximate data collection in sensor networks using probabilistic models,” in *Proc. 22nd Int. Conf. Data Eng.*, 2006, Art. no. 48.
[6] S. Asthana, O. D. King, F. D. Gibbons, and F. P. Roth, “Predicting protein complex membership using probabilistic network reliability,” *Genome Res.*, vol. 14, no. 6, pp. 1170–1175, 2004.
[7] D. Rhodes, S. Tomlins, S. Varambally, V. Mahavisno, T. Barrette, S. Kalyana-Sundaram, D. Ghosh, A. Pandey, and A. C. AM., “Probabilistic model of the human protein-protein interaction network,” *Nat. Biotechnol.*, vol. 23, no. 8, pp. 1–9, 2005.
[8] R. Jiang, Z. Tu, T. Chen, and F. Sun, “Network motif identification in stochastic networks,” *Proc. Nat. Acad. Sci. United State America*, vol. 103, no. 25, pp. 9404–9409, 2006.
[9] H. Cheng, X. Yan, and J. Han, “Mining graph patterns,” in *Proc. Frequent Pattern Mining*, 2014, pp. 307–338.
[10] Y. Cho and A. Zhang, “Predicting protein function by frequent functional association pattern mining in protein interaction networks,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 1, pp. 30–36, Jan. 2010.
[11] M. Schuhmacher and S. P. Ponzetto, “Knowledge-based graph document modeling,” in *Proc. 7th ACM Int. Conf. Web Search Data Mining*, , 2014, pp. 543–552.
[12] Y. Gu, C. Gao, L. Wang, and G. Yu, “Subgraph similarity maximal all-matching over a large uncertain graph,” *World Wide Web*, vol. 19, no. 5, pp. 755–782, 2016.
[13] Y. Yuan, G. Wang, and L. Chen, “Pattern match query in a large uncertain graph,” in *Proc. 23rd ACM Int. Conf. Inf. Knowl. Manage.*, 2014, pp. 519–528.
[14] Z. Sun, H. Wang, H. Wang, B. Shao, and J. Li, “Efficient subgraph matching on billion node graphs,” *Proc. VLDB Endowment*, vol. 5, no. 9, pp. 788–799, 2012.
[15] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, “k-nearest neighbors in uncertain graphs,” *Proc. VLDB Endowment*, vol. 3, no. 1/2, pp. 997–1008, 2010.
[16] A. Khan, F. Bonchi, A. Gionis, and F. Gullo, “Fast reliability search in uncertain graphs,” in *Proc. 17th Int. Conf. Extending Database Technol.*, 2014, pp. 535–546.
[17] R. Jin, L. Liu, B. Ding, and H. Wang, “Distance-constraint reachability computation in uncertain graphs,” *Proc. VLDB Endowment*, vol. 4, no. 9, pp. 551–562, 2011.
[18] Z. Zou, J. Li, H. Gao, and S. Zhang, “Mining frequent subgraph patterns from uncertain graph data,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 9, pp. 1203–1218, Sep. 2010.

[19] J. Li, Z. Zou, and H. Gao, "Mining frequent subgraphs over uncertain graph databases under probabilistic semantics," *Int. J. Very Large Data Bases*, vol. 21, no. 6, pp. 753–777, 2012.

[20] Y. Chen, X. Zhao, X. Lin, and Y. Wang, "Towards frequent subgraph mining on single large uncertain graphs," in *ICDM*, 2015, pp. 41–50.

[21] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proc. IEEE Int. Conf. Data Mining*, 2002, pp. 721–724.

[22] V. V. Vazirani, *Approximation Algorithms*. Berlin, Germany: Springer, 2003.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009.

[24] J. Leskovec, J. M. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 177–187.

[25] O. Papapetrou, E. Ioannou, and D. Skoutas, "Efficient discovery of frequent subgraph patterns in uncertain graph databases," in *Proc. 14th Int. Conf. Extending Database Technol.*, 2011, pp. 355–366.

[26] A. P. Mukherjee, P. Xu, and S. Tirthapura, "Mining maximal cliques from an uncertain graph," in *Proc. IEEE 31st Int. Conf. Data Eng.*, 2015, pp. 243–254.

[27] P. Hintsanen and H. Toivonen, "Finding reliable subgraphs from large probabilistic graphs," *Data Mining Knowl. Discovery*, vol. 17, no. 1, pp. 3–23, 2008.

[28] R. Jin, L. Liu, and C. C. Aggarwal, "Discovering highly reliable subgraphs in uncertain graphs," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 992–1000.

[29] G. Kollios, M. Potamias, and E. Terzi, "Clustering large probabilistic graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 2, pp. 325–336, Feb. 2013.

[30] L. Liu, R. Jin, C. C. Aggarwal, and Y. Shen, "Reliable clustering on uncertain graphs," in *Proc. IEEE 12th Int. Conf. Data Mining*, 2012, pp. 459–468.

[31] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich, "Core decomposition of uncertain graphs," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1316–1325.

[32] P. Parchas, F. Gullo, D. Papadias, and F. Bonchi, "The pursuit of a good possible world: extracting representative instances of uncertain graphs," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pp. 967–978.

[33] M. Kuramochi and G. Karypis, "Finding frequent patterns in a large sparse graph," *Data Mining Knowl. Discovery*, vol. 11, no. 3, pp. 243–271, 2005.

[34] M. Fiedler and C. Borgelt, "Support computation for mining frequent subgraphs in a single graph," in *Proc. Mining Learn. Graphs*, 2007, Art. no. 40.

[35] B. Bringmann and S. Nijssen, "What is frequent in a single graph?" in *Proc. 12th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining*, 2008, pp. 858–863.

[36] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis, "GraMi: Frequent subgraph and pattern mining in a single large graph," *Proc. VLDB Endowment*, vol. 7, no. 7, pp. 517–528, 2014.

[37] M. Kuramochi and G. Karypis, "GREW-A scalable frequent subgraph discovery algorithm," in *Proc. 4th IEEE Int. Conf. Data Mining*, 2004, pp. 439–442.

[38] C. Chen, X. Yan, F. Zhu, and J. Han, "gApprox: Mining frequent approximate patterns from a massive network," in *Proc. 7th IEEE Int. Conf. Data Mining*, 2007, pp. 445–450.

[39] S. Nijssen and J. N. Kok, "A quickstart in frequent structure mining can make a difference," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 647–652.

[40] Y. Cheng, Y. Yuan, G. Wang, B. Qiao, and Z. Wang, "Efficient sampling methods for shortest path query over uncertain graphs," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2014, pp. 124–140.

[41] Y. Tong, X. Zhang, C. C. Cao, and L. Chen, "Efficient probabilistic supergraph search over large uncertain graphs," in *Proc. 23rd ACM Int. Conf. Conf. Inf. Knowl. Manage.*, 2014, pp. 809–818.

[42] Y. Yuan, G. Wang, H. Wang, and L. Chen, "Efficient subgraph search over large uncertain graphs," *Proc. VLDB Endowment*, vol. 4, no. 11, pp. 876–886, 2011.

[43] W. Zhang, X. Lin, Y. Zhang, K. Zhu, and G. Zhu, "Efficient probabilistic supergraph search," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 4, pp. 965–978, Apr. 2016.

[44] W. E. Moustafa, A. Kimmig, A. Deshpande, and L. Getoor, "Subgraph pattern matching over uncertain graphs with identity linkage uncertainty," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, pp. 904–915.

[45] Y. Yuan, G. Wang, L. Chen, and H. Wang, "Efficient subgraph similarity search on large probabilistic graph databases," *Proc. VLDB Endowment*, vol. 5, no. 9, pp. 800–811, 2012.

[46] Z. Zou and J. Li, "Structural-context similarities for uncertain graphs," in *Proc. IEEE 13th Int. Conf. Data Mining*, 2013, pp. 1325–1330.

[47] R. Li, J. X. Yu, R. Mao, and T. Jin, "Efficient and accurate query evaluation on uncertain graphs via recursive stratified sampling," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, pp. 892–903.

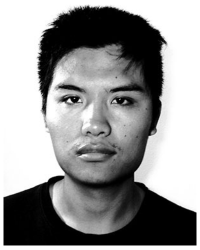
[48] A. Renyi, *Probability Theory*. North Holland, Amsterdam, The Netherlands, 1970.

[49] N. N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," *Int. J. Very Large Data Bases*, vol. 16, pp. 864–875, 2004.

Yifan Chen received the ME degree from the National University of Defense Technology (NUDT), China, in 2015. He is currently working toward the PhD degree at the NUDT, and his research interests include graph data mining and recommender systems.

Xiang Zhao received the PhD degree from The University of New South Wales, Australia, in 2014. He is currently an assistant professor with the National University of Defense Technology, China. His research interests include graph data management and mining. He is a member of the IEEE.

Xuemin Lin received the PhD degree from The University of Queensland, Australia, in 1992. He is a scientia professor with The University of New South Wales, Australia. His research interests include managing and analyzing streaming, graph, and spatial data. He is a fellow of the IEEE.



Yang Wang received the PhD degree from The University of New South Wales, Australia, in 2015. He is currently a post-doctoral researcher with The University of New South Wales. His research interests include machine learning and multimedia analytics.

Deke Guo received the PhD degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. He is currently a professor with the College of Information System and Management, National University of Defense Technology. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks. He is a senior member of the IEEE.