Recursive Stratified Sampling: A New Framework for Query Evaluation on Uncertain Graphs

Rong-Hua Li, Jeffrey Xu Yu, Rui Mao, and Tan Jin

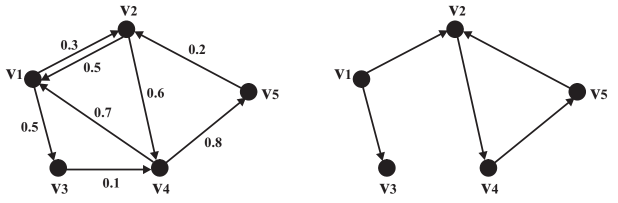
Abstract—Uncertain graph management has been recognized as an important research topic in recent years. In this paper, we first introduce two types of query evaluation problems on uncertain graphs, named expectation query evaluation and threshold query evaluation. Most previous solutions for these problems are based on naive Monte-Carlo (*NMC*) sampling, which typically result in large variances. To reduce the variance of *NMC*, we propose two efficient estimators, called *RSS-I* and *RSS-II* estimators, based on the idea of recursive stratified sampling (*RSS*). To further reduce the variances of *RSS-I* and *RSS-II*, we propose a recursive *cut-set* based stratified sampling estimator for a particular kind of query evaluation problem. We show that all the proposed estimators are unbiased and their variances are significantly smaller than that of *NMC*. Moreover, the time complexity of all the proposed estimators are the same as that of *NMC* under a mild assumption. In addition, we develop an elegant graph simplification technique to further improve the accuracy and running time of our estimators. We also apply the proposed estimators to three different uncertain graph query evaluation problems. Finally, we conduct extensive experiments to evaluate the proposed estimators, and the results show the accuracy, efficiency, and scalability of our estimators.

Index Terms—Uncertain graphs, Query evaluation, Recursive stratified sampling, Graph simplification.

1 INTRODUCTION

Uncertain graph management and mining has attracted much attention in recent years [1], [2], [3], [4]. In a widely-used uncertain graph model, each edge is associated with a probability representing the likelihood of the existence of an edge, and the existence of an edge is independent of that of any other edge [2], [3]. This model allows us to study the uncertain graph problems via the possible graph semantics [1], [2], [3]. Here a possible graph G is an instance of the uncertain graph $\mathcal{G}$, which is generated by sampling each edge in $\mathcal{G}$. Fig. 1(a) depicts an uncertain graph $\mathcal{G}$ and Fig. 1(b) illustrates a possible graph G of $\mathcal{G}$. Such an uncertain graph model is very useful to model the interaction between two nodes with uncertainty. There are many network-related applications that inherently involve uncertainty. In protein-protein interaction networks, the interaction is typically predicted by statistical models [5], [6], thereby the existence of an interaction is associated with a probability. In communication networks, the link is often associated with a failure probability [7]. In social networks, the social influence between two nodes is very often modeled by an influence probability [8], [9].

In uncertain graph management, a fundamental problem is to evaluate the queries efficiently and accurately. In this paper, we introduce two types of query evaluation problems in uncertain graphs, called expectation query evaluation and threshold query evaluation. Given an uncertain graph $\mathcal{G}$, a query q, and a query evaluation function $\phi_q(G)$ defined on the possible graph G, the expectation query evaluation problem is a problem of evaluating the expected value of $\phi_q(G)$ over all the possible graphs of $\mathcal{G}$. The threshold query evaluation is to evaluate the probability of an event that the value of $\phi_q(G)$ is greater (or less) than a given threshold δ. Many applications of uncertain graph management can be formulated as these two query evaluation problems. For instance, the classic network reliability problem [10] is an instance of the expectation query evaluation problem, where the query is a set of k nodes and the query evaluation function is a binary function used to evaluate the connectedness of the induced k-subgraph. The expected-reliable distance query problem introduced in [2] is a special instance of the expectation query evaluation problem, where the query is two given nodes and the evaluation function is the length of the shortest path between this two query nodes. The influence function evaluation problem studied in the influence maximization literature [8], [11] is also an instance of the expectation query evaluation problem, where the query is a set of seed nodes and the query evaluation function is the number of nodes that can be reachable from the seed nodes. The distance-constraint reachability problem [3] is an instance of the threshold query evaluation problem, where the query is two nodes, the threshold is the distance-constraint, and the query evaluation function is a binary function used to evaluate the reachability between two nodes subject to distance constraint.



(a) An uncertain graph $\mathcal{G}$

(b) A possible graph G of $\mathcal{G}$

Fig. 1. Illustration of an uncertain graph

In general, all the above mentioned expectation and

- Rong-Hua Li and Jeffrey Xu Yu are with The Chinese University of Hong Kong (email: {rhli, yu}@se.cuhk.edu.hk).
- Rong-Hua Li and Rui Mao are with Guangdong Province Key Laboratory of Popular High Performance Computers, Shenzhen University, China (email: lironghuascut@gmail.com, mao@szu.edu.cn).
- Tan Jin is with Northeastern University, Shenyang, China.

threshold query evaluation problems are known to be #P-complete, thus there is no polynomial algorithm to exactly solve them unless P=#P. As a result, most existing algorithms for query evaluation problems are based on naive Monte-Carlo (*NMC*) estimator [10], [2]. Specifically, the *NMC* estimator first draws N possible graphs, and then computes the query evaluation function on each possible graph. Finally, it takes the average value of the query evaluation function as the estimator. However, as discussed in [10], [3], the *NMC* estimator typically results in a large variance. Therefore, to achieve a good accuracy, the *NMC* estimator has to pick a large number of samples (possible graphs). In an uncertain graph, getting a sample needs to flip m coins to determine all the m edges of the graph. Thus, the *NMC* estimator is very expensive to obtain a good approximation for the query evaluation problems.

To reduce the variance of the *NMC* estimator, in this paper, we propose two classes of efficient estimators, named *RSS-I* and *RSS-II* estimators, based on the idea of recursive stratified sampling (*RSS*). Specifically, to obtain the *RSS-I* estimator, we first propose a basic stratified sampling estimator called *BSS-I*. *BSS-I* partitions the probability space Ω (the set of all the possible graphs) into 2^r subspaces by enumerating all the statuses (0 or 1) of r selected edges[1]. Let each subspace be a stratum. Then, *BSS-I* draws samples separately from each stratum. By carefully allocating the sample size for each stratum, we show that the variance of the *BSS-I* estimator is smaller than that of *NMC*. Based on *BSS-I*, we develop the *RSS-I* estimator by recursively applying *BSS-I* in each stratum. Since *RSS-I* recursively reduces the variance in each stratum, its variance is significantly smaller than that of *BSS-I*. Similarly, to obtain the *RSS-II* estimator, we also propose a new basic stratified sampling estimator, called *BSS-II* estimator, and then recursively apply *BSS-II* in each stratum. Unlike *BSS-I* and *RSS-I*, we develop a new stratification method for *BSS-II* and *RSS-II* which splits the probability space Ω into $r+1$ strata by selecting r edges. Compared to the stratification method used in *BSS-I* and *RSS-I*, the advantage of this method is that we can finely tune the number of strata by parameter r, because it only produces $r+1$ strata. We show that all these estimators are unbiased and their variances are significantly smaller than that of *NMC*. In addition, an important feature of our estimators is that they have the same time complexity as *NMC* under a mild assumption, satisfying in most real-world applications.

The proposed estimators are very general which can be used as a building block for most query evaluation on uncertain graph problems. However, these methods do not exploit the graph structure information and the property of the query evaluation function as well. To capture these information, we further propose a basic *cut-set* based stratified sampling estimator and a recursive *cut-set* based stratified sampling estimator, called *BCSS* and *RCSS* respectively, for a particular kind of query evaluation problem where the query evaluation function has a *cut-set* property. The detailed definition of *cut set* can be found in Section 5. We prove that both *BCSS* and *RCSS* are unbiased and their variances are significantly smaller than that of *NMC*. Furthermore, in many applications, we show that the time complexity of *BCSS* and *RCSS* estimators are the same as that of *NMC*. In addition, we integrate a new graph simplification technique into all the proposed estimators, and show that the graph simplification technique not only significantly improves the accuracy, but it also shortens the running time of our estimators. We also apply the proposed estimators to the influence function evaluation, the expected-reliable distance query, as well as the distance-constraint reachability query, which are the instances of our query evaluation problems. Finally, we conduct extensive experiments to evaluate the proposed estimators. The results show that the proposed recursive stratified estimators with graph simplification technique significantly outperform the state-of-the-art estimator. Moreover, we find that the *RCSS* estimators can improve the accuracy over both *RSS-I* and *RSS-II* estimators. The results also show that our best estimators scale linearly w.r.t. the graph size, thus they can be used to handle large graphs.

The rest of this paper is organized as follows. We formulate the query evaluation problems in Section 2. We propose the *RSS-I* and *RSS-II* estimators with graph simplification technique in Section 3 and Section 4 respectively. The *BCSS* and *RCSS* estimators are presented in Section 5. We apply our estimators with graph simplification technique to three different uncertain graph queries in Section 6, and report the experimental results in Section 7. We discuss the related work in Section 8. Finally, we conclude this work in Section 9.

2 Problem formulation

Consider an uncertain graph $\mathcal{G} = (V, E, P)$ with $|V| = n$ and $|E| = m$, where V and E denote the set of nodes and edges respectively. P is a set of probabilities representing the likelihoods of the existence of edges, i.e., p_e denotes the probability of $e \in E$. In this paper, we adopt a widely-used uncertain graph model where the existence of an edge is independent of that of any other edge [2], [3]. Let $G = (V_G, E_G)$ be a possible graph which is obtained by sampling each edge e in $\mathcal{G}$ following the probability p_e. Obviously, $V = V_G$, $E_G \subseteq E$, and the probability of G is given by

$$\Pr[G] = \prod\nolimits_{e \in E_G} p_e \prod\nolimits_{e \in E \setminus E_G} (1 - p_e). \tag{1}$$

Consider an example in Fig. 1. Fig. 1(a) shows an uncertain graph with 5 nodes and 8 edges and Fig. 1(b) illustrates a possible graph G of $\mathcal{G}$ with probability 0.001944.

Given an uncertain graph $\mathcal{G}$, a query q, and a query evaluation function $\phi_q(G)$ defined on the possible graph G. We define two types of query evaluation problems as follows.

Definition 2.1: (Expectation query evaluation) The expectation query evaluation problem is a problem of computing the expected value of $\phi_q(G)$ over all the possible graphs, which is given by

$$\Phi_q(\mathcal{G}) = \sum\nolimits_{G \in \Omega} \Pr[G]\phi_q(G), \tag{2}$$

where Ω denotes the set of all possible graphs of $\mathcal{G}$.

Definition 2.2: (Threshold query evaluation) Given a threshold δ, the threshold query evaluation problem is a problem of calculating the following expected value

$$\mathbb{I}_q(\mathcal{G}) = \sum\nolimits_{G \in \Omega} \Pr[G]\mathrm{I}_q(G), \tag{3}$$

[1] Here the status of an edge is 1 denoting the edge exists in the possible graph, and 0 otherwise. Thus, for r edges, there are 2^r different cases.

where $\mathrm{I}_q(G)$ is an indicator variable. More specifically, $\mathrm{I}_q(G)$ is defined by

$$\mathrm{I}_q(G) = \begin{cases} 1, & if\ \mathscr{C}(\phi_q(G),\delta) = 1 \\ 0, & otherwise \end{cases}, \tag{4}$$

where $\mathscr{C}(\phi_q(G),\delta)$ is a binary comparison function.

Note that the comparison function $\mathscr{C}(\phi_q(G),\delta)$ in Eq. (4) is used to compare the query evaluation function $\phi_q(G)$ with the given threshold δ. For example, the comparison function can be a "$\leq$" function, i.e., $\mathscr{C}(\phi_q(G),\delta) = 1$ if $\phi_q(G) \leq \delta$, and $\mathscr{C}(\phi_q(G),\delta) = 0$ otherwise.

As discussed in Section 1, many uncertain graph problems, such as network reliability estimation [10], expected-reliable distance query [2], distance-constraint reachability computation [3] and influence function evaluation [8], [11], can be formulated as the above two query evaluation problems. In general, all aforementioned query evaluation problems are known to be #P-complete, thus there does not exist a polynomial algorithm to exactly solve them unless P=#P. Hence, most existing algorithms for these problems are based on a naive Monte-Carlo (*NMC*) estimator [10], [2], [8], [11]. In the rest of this paper, we mainly focus on the expectation query, and similar techniques can be easily generalized to the threshold query.

To estimate the expectation query $\Phi_q(\mathcal{G})$, the *NMC* algorithm first draws N possible graphs denoted by $G_1, G_2, \cdots, G_N$ from $\mathcal{G}$. Then, for each possible graph G_i, the *NMC* algorithm calculates the query evaluation function $\phi_q(G_i)$. Finally, the *NMC* estimator (denoted by $\hat{\Phi}_{NMC}$) is obtained by taking the mean of $\phi_q(G_i)$ $(i = 1, 2, \cdots, N)$, i.e., $\hat{\Phi}_{NMC} = \sum_{i=1}^{N} \phi_q(G_i)/N$. The *NMC* estimator is unbiased and its variance is given by

$$\mathrm{var}(\hat{\Phi}_{NMC}) = [\textstyle\sum_{G_P \in \Omega} \Pr[G_P]\phi_q(G)^2 - \Phi_q(\mathcal{G})^2]/N. \tag{5}$$

Assume that computing $\phi_q(G_i)$ takes $O(M)$ time. Then, we can easily derive that the time complexity of *NMC* is $O(N(m + M))$.

An important metric to evaluate the accuracy of the Monte-Carlo based algorithm is the mean squared error (MSE) which is denoted by $\mathbb{E}[(\hat{\Phi}_q(\mathcal{G}) - \Phi_q(\mathcal{G}))^2]$, where $\hat{\Phi}_q(\mathcal{G})$ denotes an estimator of $\Phi_q(\mathcal{G})$ by the Monte-Carlo based algorithm. By the so-called variance-bias decomposition [3], this metric can be decomposed into two parts.

$$\mathbb{E}[(\hat{\Phi}_q(\mathcal{G}) - \Phi_q(\mathcal{G}))^2] = \mathrm{var}[\hat{\Phi}_q(\mathcal{G})] + [\mathbb{E}[\hat{\Phi}_q(\mathcal{G})] - \Phi_q(\mathcal{G})]^2, \tag{6}$$

where $\mathbb{E}[\hat{\Phi}_q(\mathcal{G})]$ and $\mathrm{var}[\hat{\Phi}_q(\mathcal{G})]$ denote the expectation and variance of the estimator $\hat{\Phi}_q(\mathcal{G})$ respectively. If the estimator is unbiased, then the second term in Eq. (6) will be vanished. Therefore, the variance of an unbiased estimator is the only indicator for evaluating the accuracy of the estimator.

As discussed in [10], [3], the *NMC* estimator typically results in a large variance, which significantly reduces its accuracy. An effective approach to improve the accuracy of *NMC* is to reduce its variance. To that end, in the following sections, we shall propose several new estimators based on stratified sampling [12] without sacrificing efficiency.

3 New Class-I Estimators

To reduce the variance of *NMC*, in this section, we propose two stratified sampling estimators for expectation query evaluation, called *BSS-I* and *RSS-I* estimators. The *RSS-I* estimator is based on the idea of recursive stratified sampling which recursively applies *BSS-I* in each stratum. For convenience, we refer to both *BSS-I* and *RSS-I* as the class-I estimators. Below, we first introduce the *BSS-I* estimator, followed by the *RSS-I* estimator.

TABLE 1
Stratum design of class-I estimators

Edges	e_1	e_2	e_3	$\cdots$	e_r	e_{r+1}	$\cdots$	e_m	Prob. space
Stratum 1	0	0	0	$\cdots$	0	$*$	$\cdots$	$*$	Ω_1
Stratum 2	1	0	0	$\cdots$	0	$*$	$\cdots$	$*$	Ω_2
Stratum 3	0	1	0	$\cdots$	0	$*$	$\cdots$	$*$	Ω_3
$\cdots$				$\cdots$					$\cdots$
Stratum 2^r	1	1	1	$\cdots$	1	$*$	$\cdots$	$*$	Ω_{2^r}

3.1 Basic stratified sampling (*BSS-I*)

Unlike *NMC* which draws a sample (a possible graph) from the entire population (all the possible graphs), the stratified sampling method [12] first divides the population into several disjoint groups called *strata*, and then independently picks separate samples from these groups. As a commonly used technique for reducing variance in sampling design [12], there are two key technical challenges in stratified sampling: *stratification*, which is a process for partitioning the entire population into disjoint strata, and *sample allocation*, which is a procedure to determine the sample size that needs to be drawn from each stratum. Below, we will present our stratification and sample allocation methods.

Stratification. Let e_i $(i = 1, \cdots, m)$ be an edge in an uncertain graph $\mathcal{G}$. First, we choose r edges $(e_1, \cdots, e_r)$ and determine their statuses (0 or 1), where r is a small number. For the rest $m - r$ edges, we set their statuses to $*$ which means that "their statuses are undetermined". Note that this process partitions the entire probability space Ω (i.e., the set of all possible graphs) into 2^r subspaces $\Omega_1, \cdots, \Omega_{2^r}$. Second, we let each subspace be a stratum. This is because $\Omega_1, \cdots, \Omega_{2^r}$ are disjoint sets and $\Omega = \bigcup_{i=1}^{2^r} \Omega_i$, thus each subspace is indeed a valid stratum. The idea of our stratification method is illustrated in Table 1.

Let $T = (e_1, e_2, \cdots, e_r)$ be the set of selected r edges, and $X_i = (X_{i,1}, X_{i,2}, \cdots, X_{i,r})$ be the status vector corresponding to the selected r edges in Stratum i, where $X_{i,j} = 0$ represents that the edge e_j is failed, and $X_{i,j} = 1$ denotes that the edge e_j exists. For example, for Stratum 1 in Table 1, the status vector is $X_1 = (0, 0, \cdots, 0)$, which means that all the selected r edges are failed. In other words, all the possible graphs in Ω_1 do not include the edges in T. The probability of a possible graph in Stratum i $(i = 1, \cdots, 2^r)$ is given by

$$\pi_i = \Pr[G_P \in \Omega_i] = \prod_{e_j \in T \wedge X_{i,j}=1} p_j \prod_{e_j \in T \wedge X_{i,j}=0} (1 - p_j). \tag{7}$$

In our stratification approach, a question that arises is how to select the r edges for stratification. As shown in the previous version of this paper [13], the edge-selection strategy for choosing r edges significantly affects the performance of the estimator. One straightforward strategy is to randomly pick r edges from the edge set E. We refer to this edge-selection strategy as the random edge-selection strategy (*RM*). With the *RM* strategy, the selected r edges may not directly contribute to compute the query evaluation function $\phi_q(G)$. For example, assume that the query evaluation function $\phi_q(G)$ denotes the number of nodes in the possible graph G that are reachable from the query node q (i.e., this query evaluation problem is an instance of influence function evaluation problem [8], [11]). Further, we suppose that the uncertain graph $\mathcal{G}$ has two connected components and the query node q is in the first component. If all the selected r edges are in the second component, then these r edges make no contribution to compute $\phi_q(G)$. This may reduce the performance of *BSS-I*. To avoid such a problem, we introduce a heuristic edge-selection strategy based on the *BFS* (breadth-first-search) visiting order of the edges. To estimate $\Phi_q(\mathcal{G})$, we first invoke a *BFS* algorithm starting from the query node q to obtain the first r edges according to the *BFS* visiting order of the edges. Then, we use these r edges for stratification. We refer to such edge-selection strategy as the *BFS* edge-selection strategy. Obviously, according to the *BFS* strategy, the selected edges directly contribute to computing $\phi_q(G)$. It is important to emphasize that the *RM* strategy is very general which can be used for any query evaluation problems, while the *BFS* edge-selection strategy only work well on a class of query evaluation problems where the query evaluation function can be calculated by the *BFS* algorithm, such as the reachability query [3], shortest path query [2], network reliability estimation [10], and influence function evaluation [8].

The *BSS-I* estimator. Let N be the total number of samples, N_i be the number of samples drawn from Stratum i ($i = 1, 2, \cdots, 2^r$), and $G_{i,j}$ ($j = 1, 2, \cdots, N_i$) be a possible graph sampled from Stratum i. Then, *BSS-I* is given as follows.

$$\hat{\Phi}_{BSSI} = \sum\nolimits_{i=1}^{2^r} \pi_i \frac{1}{N_i} \sum\nolimits_{j=1}^{N_i} \phi_q(G_{i,j}), \tag{8}$$

where π_i is defined in Eq. (7). The following theorem shows that $\hat{\Phi}_{BSSI}$ is an unbiased estimator of $\Phi_q(\mathcal{G})$. All the missing proofs of this paper can be found in the supplementary document of the paper.

Theorem 3.1: $\mathbb{E}(\hat{\Phi}_{BSSI}) = \Phi_q(\mathcal{G})$.

Let σ_i be the variance of the sample in Stratum i. Since the samples are independently drawn by the basic stratified sampling algorithm, thus the variance of *BSS-I* is

$$\mathrm{var}(\hat{\Phi}_{BSSI}) = \sum\nolimits_{i=1}^{2^r} \pi_i^2 \sigma_i / N_i. \tag{9}$$

Sample allocation. As shown in Eq. (9), the variance of *BSS-I* depends on the sample size of all strata, i.e., N_i, for $i = 1, 2, \cdots, 2^r$. Thus, the question is how to allocate the sample size for each Stratum i ($i = 1, 2, \cdots, 2^r$) so as to minimize the variance of *BSS-I*, i.e., $\mathrm{var}(\hat{\Phi}_{BSSI})$. Formally, the sample allocation problem is formulated as follows.

$$\begin{array}{ll} \min & \mathrm{var}(\hat{\Phi}_{BSSI}) = \sum_{i=1}^{2^r} \pi_i^2 \frac{\sigma_i}{N_i} \\ s.t. & \sum_{i=1}^{2^r} N_i = N. \end{array} \tag{10}$$

By applying the Lagrangian method, we can derive the optimal sample allocation strategy which is given by

$$N_i = N\pi_i\sqrt{\sigma_i} / \sum\nolimits_{i=1}^{2^r} \pi_i\sqrt{\sigma_i}, \tag{11}$$

for $i = 1, \cdots, 2^r$. From Eq. (11), the optimal allocation needs to know the variance of the sample in each stratum, i.e. σ_i, for $i = 1, \cdots, 2^r$. Unfortunately, such variances are unavailable in our problem. However, if we set the sample size of Stratum i to $\pi_i N$ (proportional sample allocation), then the variance of *BSS-I* will be no larger than the variance of *NMC*.

Algorithm 1 *BSS-I* ($\mathcal{G}$, N, q, r)

```
Input: An uncertain graph $\mathcal{G} = (V, E, P)$, the sample size $N$,
       a query $q$, and the stratification parameter $r$.
Output: The BSS-I estimator $\hat{\Phi}$.

 1: $\hat{\Phi} \leftarrow 0$;
 2: Choose $r$ edges from $E$ by an edge-selection strategy;
 3: for $i = 1$ to $2^r$ do
 4:     Let $X_i$ be the status vector of Stratum $i$ (Table 1);
 5:     $\mathcal{G}_i \leftarrow$ simplify graph $\mathcal{G}$ based on $X_i$;
 6:     Compute $\pi_i$ by Eq. (7), $N_i \leftarrow [\pi_i N]$, $t \leftarrow 0$;
 7:     for $j = 1$ to $N_i$ do
 8:         Sampling $\mathcal{G}_i$ to generate a possible graph $G_j$;
 9:         Compute $\phi_q(G_j)$, $t \leftarrow t + \phi_q(G_j)$;
10:     $t \leftarrow t/N_i$, $\hat{\Phi} \leftarrow \hat{\Phi} + \pi_i t$;
11: return $\hat{\Phi}$;
```

Theorem 3.2: *If* $N_i = \pi_i N$, $\mathrm{var}(\hat{\Phi}_{BSSI}) \leq \mathrm{var}(\hat{\Phi}_{NMC})$.

Equipped with the stratification and sample allocation methods, we can easily devise the *BSS-I* algorithm. The detailed implementation of the *BSS-I* algorithm can be found in the preliminary version of this paper [13]. Below, we propose a new implementation which integrates an elegant graph simplification technique into the *BSS-I* algorithm.

The graph simplification technique. The key idea of the graph simplification technique is that we are able to simplify the uncertain graph after determining the statuses of the selected r edges in the *BSS-I* algorithm. This is because in the stratification procedure (Table 1), some edges are definitely failed in strata from 1 to $2^r - 1$. In each of these strata, the failed edges (statuses are 0) may lead to many other edges that are irrelevant to the query. Therefore, all those irrelevant edges can be pruned, thus simplifying the original uncertain graph. For example, let us consider the influence function evaluation problem where the query can be a node s and the goal is to compute the expected number of nodes that are reachable from s. In this example, after deleting the failed edges, all the residual edges that cannot be traversed by a *BFS* starting from s are irrelevant edges. As a result, we can prune all those edges because they are irrelevant to the query.

The general framework of the *BSS-I* algorithm with graph simplification technique is outlined in Algorithm 1. Unlike our previous implementation [13], the new implementation includes a graph simplification step (line 5), where we simplify the uncertain graph after determining the status vector X_i for each stratum i. After that, we perform the same stratified sampling procedure as [13] on the simplified uncertain graph. It should be noted that in Algorithm 1, we give a general framework with the graph simplification technique (line 5), the detailed implementation of the graph simplification procedure is a nontrivial task, which depends on different applications. In Section 6, we will show how to implement the graph simplification techniques for three different uncertain graph applications.

The main advantages of the graph simplification technique are threefold. First, it reduces the graph size, thus clearly saving the computational cost for sampling the uncertain graph and calculating the query evaluation function as well. Second, it eliminates the "uncertainty" of the uncertain graph after pruning the irrelevant edges, thus it is expected to improve the accuracy of the estimators. Indeed, in our experiments, we will show that with the graph simplification technique, the new implementations of our algorithms not only shorten the running time, but also significantly improve the accuracy of the algorithms. Third, we will show that for the recursive stratified sampling algorithms, the graph simplification techniques can avoid to pick the irrelevant edges for stratification, because the irrelevant edges are pruned in each stratum.

For the time complexity of Algorithm 1, we assume that computing the query evaluation function for each possible graph takes $O(M)$ time (line 9). Further, we assume that the total time overhead of the graph simplification procedure can be dominated by $O(m)$. In Section 6, we will show how to achieve this in three different applications. Then, under these assumption, we can easily derive that the time complexity of Algorithm 1 is $O(N(m+M))$ which is equal to that of the *NMC* algorithm.

3.2 Recursive stratified sampling (*RSS-I*)

Recall that *BSS-I* splits the entire set of possible graphs into 2^r subsets. We observe that *BSS-I* can be recursively applied into any subsets. Based on this observation, we develop a recursive stratified sampling estimator, called *RSS-I*. The original implementation of the *RSS-I* algorithm can be found in [13]. Here we present an optimized implementation given in Algorithm 2, which integrates the graph simplification technique into the recursive stratified sampling procedure. Specifically, *RSS-I* recursively partitions the sample size N to $N_i = \pi_i N$ $(i = 1, 2, \cdots, 2^r)$ to estimate $\Phi_q(\mathcal{G})$ in Stratum i (lines 9-15 in Algorithm 2). Moreover, in each Stratum i, *RSS-I* recursively simplifies the uncertain graph (line 12). The *RSS-I* algorithm terminates until the sample size is smaller than a given threshold (τ) or the number of residual uncertain edges is smaller than r (line 2). When the terminative conditions of the *RSS-I* algorithm are satisfied, we perform a naive Monte-Carlo sampling to estimate $\Phi_q(\mathcal{G})$ (lines 3-7). Similar to *BSS-I*, the partition approach in *RSS-I* also relies on the edge-selection strategy (line 9). Likewise, for the general query evaluation problem, we can use the random edge-selection (*RM*) strategy. For the query evaluation problems in which the query evaluation function can be solved by the *BFS* algorithm, we recommend to use the *BFS* edge-selection strategy. Similarly, suppose that the total time complexity for the graph simplification procedure is dominated by $O(m)$. Then, by a similar analysis shown in [13], we can derive that the time complexity of *RSS-I* is $O(N(m+M))$ which is the same as the *BSS-I* and *NMC* algorithms.

Since *BSS-I* is unbiased, *RSS-I* is also unbiased. Moreover, *RSS-I* reduces the variance in each partition, thus the variance of *RSS-I* is no larger than the variance of *BSS-I*. Formally, we have the following theorem.

Theorem 3.3: *Let* $\mathrm{var}(\hat{\Phi}_{RSSI})$ *be the variance of RSS-I, then* $\mathrm{var}(\hat{\Phi}_{RSSI}) \leq \mathrm{var}(\hat{\Phi}_{BSSI})$.

Relation to the algorithm proposed in [3]. In [3], the authors propose a similar recursive stratified sampling method for the distance-constraint reachability query problem in an uncertain graph. Their algorithm recursively picks one edge for stratification. Specifically, in a stratification, their algorithm partitions the population into two sub-populations, and then recursively applies the same idea into each sub-population. Essentially, their algorithm is a special case of the *RSS-I* algorithm when $r = 1$. Similarly, their algorithm can also be integrated with the graph simplification technique proposed in this paper. The main drawback of their method is that when $r = 1$, the effectiveness of the graph simplification technique is rather limited, because it generally cannot prune too many irrelevant edges based on only one failed edge. However, the *RSS-I* algorithm can alleviate this issue, because we can choose an appropriate r ($r > 1$) to achieve a good graph simplification performance. In addition, different from their algorithm, in the following sections, we will propose two new recursive stratification sampling algorithms based on drastically different stratification techniques, which are more flexible than the *RSS-I* algorithm.

Algorithm 2 *RSS-I* ($\mathcal{G}$, N, q, r, τ)

```
Input: An uncertain graph $\mathcal{G} = (V, E, P)$, the sample size $N$,
       a query $q$, and parameters $r$ and $\tau$
Output: The RSS-I estimator $\hat{\Phi}$.

 1: $\hat{\Phi} \leftarrow 0$;
 2: if $N < \tau$ or $|E| < r$ then
 3:     for $j = 1$ to $N$ do
 4:         Sampling $\mathcal{G}$ to generate a possible graph $G_j$;
 5:         Compute $\phi_q(G_j)$;
 6:         $\hat{\Phi} \leftarrow \hat{\Phi} + \phi_q(G_j)$;
 7:     return $\hat{\Phi}/N$;
 8: else
 9:     $T \leftarrow$ select $r$ edges from $\mathcal{G}$ by an edge-selection strategy;
10:     for $i = 1$ to $2^r$ do
11:         Let $X_i$ be the status vector of set $T$ in Stratum $i$
            (Table 1);
12:         $\mathcal{G}_i \leftarrow$ simplify graph $\mathcal{G}$ based on $X_i$;
13:         Compute $\pi_i$ by Eq. (7), $N_i \leftarrow [\pi_i N]$;
14:         $\mu_i \leftarrow$ RSS-I ($\mathcal{G}_i$, $N_i$, $q$, $r$, $\tau$);
15:         $\hat{\Phi} \leftarrow \hat{\Phi} + \pi_i \mu_i$;
16:     return $\hat{\Phi}$;
```

4 New Class-II Estimators

In this section, we propose two new stratified sampling estimators, named *BSS-II* and *RSS-II* estimators. The *BSS-II* estimator is based on a new stratification method, which allows us to finely tune the number of strata, thus it is more flexible than *BSS-I* in practice. Similar to *RSS-I*, the *RSS-II* estimator uses the idea of recursive stratified sampling, which recursively applies *BSS-II* in each stratum. To distinguish the class-I estimators, we refer to both *BSS-II* and *RSS-II* as the class-II estimators.

4.1 Basic stratified sampling (*BSS-II*)

Stratification. In *BSS-II*, the new stratification method splits the probability space Ω into $r+1$ various subspaces $(\Omega_0, \cdots, \Omega_r)$ by choosing r edges. Specifically, for Stratum 0, we set the statuses of all the r selected edges to "0", and for Stratum i $(i \neq 0)$, we set the status of edge i to "1", the statuses of all the previous $i-1$ edges (i.e., $e_1, \cdots, e_{i-1}$) to "0", and the statuses of the rest of edges to "$*$" denoting that their statuses are undetermined. Unlike the

TABLE 2
Stratum design of class-II estimators

Edges	e_1	e_2	e_3	$\cdots$	e_r	e_{r+1}	$\cdots$	e_m	Prob. space
Stratum 0	0	0	0	$\cdots$	0	$*$	$\cdots$	$*$	Ω_0
Stratum 1	1	$*$	$*$	$\cdots$	$*$	$*$	$\cdots$	$*$	Ω_1
Stratum 2	0	1	$*$	$\cdots$	$*$	$*$	$\cdots$	$*$	Ω_2
Stratum 3	0	0	1	$\cdots$	$*$	$*$	$\cdots$	$*$	Ω_3
$\cdots$				$\cdots$					$\cdots$
Stratum r	0	0	0	$\cdots$	1	$*$	$\cdots$	$*$	Ω_r

stratification method used in *BSS-I*, this new stratification approach allows us to finely tune the stratification parameter r because it only generates $r+1$ strata. However, for the *BSS-I* estimator, r leads to 2^r strata, thus it is hard to tune the number of strata of the estimator. Moreover, compare to the stratification method used in *BSS-I*, we will show in Section 6 that this new stratification method is relatively easy to integrate with the graph simplification technique. The new stratum design method is depicted in Table 2.

In Table 2, each stratum (Stratum 0, Stratum 1, $\cdots$, Stratum r) corresponds to a subspace $(\Omega_0, \Omega_1, \cdots, \Omega_r)$. For any $i \neq j$, we have $\Omega_i \cap \Omega_j = \emptyset$. Below, we show that $\bigcup_{i=0}^{r} \Omega_i = \Omega$. Let $T = (e_1, e_2, \cdots, e_r)$ be the set of r selected edges and p_j $(j = 1, \cdots, r)$ be the corresponding probability, then the probability of a possible graph in Stratum i is given by

$$\pi'_i = \Pr[G_P \in \Omega_i] = \begin{cases} \prod_{j=1}^{r} (1-p_j), & if\ i = 0 \\ p_i \prod_{j=1}^{i-1} (1-p_j), & otherwise \end{cases} \tag{12}$$

The following theorem implies that $\bigcup_{i=0}^{r} \Omega_i = \Omega$.

Theorem 4.1: $\sum_{i=0}^{r} \Pr[G_P \in \Omega_i] = 1$.

By Theorem 4.1, the stratum design approach described in Table 2 is a valid stratification method.

The *BSS-II* estimator. Similar to *BSS-I*, we let N be the total sample size, and N_i be the sample size of Stratum i, and $G_{i,j}$ $(j = 1, 2, \cdots, N_i)$ be a possible graph sampled from Stratum i. Then, the *BSS-II* estimator is given by

$$\hat{\Phi}_{BSSII} = \sum\nolimits_{i=0}^{r} \pi'_i \frac{1}{N_i} \sum\nolimits_{j=1}^{N_i} \phi_q(G_{i,j}), \tag{13}$$

where π'_i is given in Eq. (12). Similar to Theorem 3.1, the following theorem shows that *BSS-II* is unbiased.

Theorem 4.2: $\mathbb{E}(\hat{\Phi}_{BSSII}) = \Phi_q(\mathcal{G})$.

The variance of *BSS-II* is given by

$$\mathrm{var}(\hat{\Phi}_{BSSII}) = \sum\nolimits_{i=0}^{r} {\pi'_i}^2 \sigma_i / N_i, \tag{14}$$

where σ_i denotes the variance of the sample in Stratum i.

Sample allocation. Similar to the sample allocation approach used in *BSS-I*, for *BSS-II*, we set the sample size of Stratum i to $\pi'_i N$, i.e., $N_i = \pi'_i N$. Based on this sample allocation method, we show that the variance of *BSS-II* is no larger than the variance of *NMC*.

Theorem 4.3: *If* $N_i = \pi'_i N$, $\mathrm{var}(\hat{\Phi}_{BSSII}) \leq \mathrm{var}(\hat{\Phi}_{NMC})$.

With the stratification and sample allocation methods, we are ready to give the *BSS-II* algorithm. Likewise, the graph simplification technique used in *BSS-I* can also be applied to *BSS-II*. The detailed description of the *BSS-II* algorithm with the graph simplification technique can be found in the supplementary document of this paper. It is worth mentioning that the edge-selection strategies used in *BSS-I* can also be used in *BSS-II*. In addition, we can also derive that the time complexity of *BSS-II* is $O(N(m+M))$ based on the same assumption as made in *BSS-I*.

4.2 Recursive stratified sampling (*RSS-II*)

Based on *BSS-II*, we develop another new recursive stratified sampling estimator (*RSS-II*). Similar to the idea of *RSS-I*, *RSS-II* makes use of *BSS-II* as a building block and recursively applies *BSS-II* in each stratum. More specifically, *RSS-II* first partitions the probability space Ω into $r+1$ subspace Ω_i $(i = 0, 1, \cdots, r)$ by the stratification method used in *BSS-II*. Then, the same partition procedure is recursively performed in each subspace Ω_i. In each partition, *RSS-II* utilizes the sample allocation method used in *BSS-II* to allocate the sample size. Similarly, the graph simplification technique is also recursively applied in each stratum. The recursion process of *RSS-II* will terminate until the sample size is smaller than a given threshold (τ) or the number of residual uncertain edges is smaller than r. Likewise, *RSS-II* is unbiased and its variance is no larger than that of *BSS-II*. Also, we can derive that the time complexity of the *RSS-II* algorithm is $O(N(m+M))$ by assuming that the total time complexity of the graph simplification procedure is bounded by $O(m)$. We will show how to achieve this time complexity in Section 6 for three different uncertain graph query applications. The detailed description of the *RSS-II* algorithm can be found in the supplementary document of this paper.

5 Cut-set based estimators

In this section, we propose two cut-set based estimators to further improve the accuracy of our class-I and class-II estimators for a kind of problem where the query evaluation function has a so-called *cut-set* property. Below, we first present a new sampling algorithm called *focal sampling* which forms a building block for developing the cut-set based estimators.

5.1 Focal sampling

First, we define the *cut set* for a query evaluation function.

Definition 5.1: Given an uncertain graph $\mathcal{G} = (V, E, P)$, a query q, and a query evaluation function $\phi_q(G)$, the cut set C is a strict subset of edges (i.e., $C \subset E$) such that $\phi_q(G)$ is a constant for all the possible graphs G with all edges in C are failed (i.e., their statuses are zeros).

A query evaluation function $\phi_q(G)$ has a cut-set property if and only if there is a cut set C satisfying the above definition. According to Definition 5.1, for a query q, we can partition the probability space Ω into two subspaces denoted by Ω_0 and $\bar{\Omega}_0$ based on the cut set C. Here Ω_0 is a set of all the possible graphs such that for any possible graph $G = (V_G, E_G) \in \Omega_0$ we have $E_G \cap C = \emptyset$, and $\bar{\Omega}_0 = \Omega \backslash \Omega_0$. Let $C = \{e_1, e_2, \cdots, e_{|C|}\}$, then the probability of a possible graph G in Ω_0 is given by

$$\Pr[G \in \Omega_0] = \pi_0^c = \prod\nolimits_{j=1}^{|C|} (1-p_j), \tag{15}$$

and the probability of $G \in \bar{\Omega}_0$ is given by $\bar{\pi}_0^c = 1 - \pi_0^c$. Assume that for any possible graph $G \in \Omega_0$, $\phi_q(G) = u_0$ is a constant and can be easily calculated. Then, to evaluate $\Phi_q(\mathcal{G})$, we do not need to draw samples from Ω_0. Instead, we can focus on picking samples from $\bar{\Omega}_0$. Based on this

TABLE 3
Stratum design of cut-set based estimators

Edges	e_1	e_2	e_3	$\cdots$	$e_{\lvert C\rvert}$	$e_{\lvert C\rvert+1}$	$\cdots$	e_m	Prob. Space
Stratum 1:	1	$*$	$*$	$\cdots$	$*$	$*$	$\cdots$	$*$	Ω_1
Stratum 2:	0	1	$*$	$\cdots$	$*$	$*$	$\cdots$	$*$	Ω_2
Stratum 3:	0	0	1	$\cdots$	$*$	$*$	$\cdots$	$*$	Ω_3
$\cdots$				$\cdots$					$\cdots$
Stratum $\lvert C\rvert$:	0	0	0	$\cdots$	1	$*$	$\cdots$	$*$	$\Omega_{\lvert C\rvert}$

idea, we propose the focal sampling (FS) estimator $\hat{\Phi}_{FS}$ as follows

$$\hat{\Phi}_{FS} = \pi_0^c u_0 + \bar{\pi}_0^c \sum\nolimits_{i=1}^{N} \phi_q(G_i)/N, \tag{16}$$

where N is the sample size and G_i is a possible graph drawn from $\bar{\Omega}_0$. The following theorem shows that $\hat{\Phi}_{FS}$ is an unbiased estimator of $\Phi_q(\mathcal{G})$.

Theorem 5.1: $\mathbb{E}(\hat{\Phi}_{FS}) = \Phi_q(\mathcal{G})$.

Let $\bar{u}_0$ and $\bar{\sigma}_0$ be the expectation and variance of the samples in $\bar{\Omega}_0$ respectively. Theorem 5.2 shows that the variance of the FS estimator ($\hat{\Phi}_{FS}$) is no larger than that of the *NMC* estimator.

Theorem 5.2: $\mathrm{var}(\hat{\Phi}_{FS}) \le \mathrm{var}(\hat{\Phi}_{NMC})$.

5.2 The *BCSS* estimator

Recall that in the FS estimator, we need to draw samples from $\bar{\Omega}_0$ by *NMC*. To further reduce its variance, we propose a basic cut-set based stratified sampling estimator, called *BCSS*, which uses stratified sampling to draw samples from $\bar{\Omega}_0$. Similarly, there are two key techniques in *BCSS*: stratification and sample allocation. Below, we first present the stratification method, and then describe the sample allocation strategy.

Stratification. First, we divide the probability space $\bar{\Omega}_0$ into $|C|$ subspaces based on the cut set C, which is denoted by $\Omega_1, \cdots, \Omega_{|C|}$. Then, we let each subspace to be a stratum, i.e., Ω_i denotes Stratum i for $i = 1, \cdots, |C|$, and draw samples separately from each stratum. The detailed stratification method is given in Table 3.

Based on the stratification method, we can easily derive that for any $i \neq j$, we have $\Omega_i \cap \Omega_j = \emptyset$ and $\bigcup_{i=1}^{|C|} \Omega_i = \bar{\Omega}_0 = \Omega \backslash \Omega_0$. The probability of a possible graph in Stratum i is given by

$$\Pr[G \in \Omega_i] = \pi_i^c = p_i \prod\nolimits_{j=1}^{i-1} (1 - p_j), \tag{17}$$

where $i = 1, \cdots, |C|$. Also, it is easy to show that

$$\sum\nolimits_{i=1}^{|C|} \pi_i^c = 1 - \pi_0^c, \tag{18}$$

where π_0^c is given in Eq. (15).

The estimator. Let N be the sample size, N_i be the sample size of Stratum i, and $G_{i,j}$ be a possible graph drawn from Stratum i. Then, the *BCSS* estimator is given by

$$\hat{\Phi}_{BCSS} = \pi_0^c u_0 + \sum\nolimits_{i=1}^{|C|} \pi_i^c \sum\nolimits_{j=1}^{N_i} \phi_q(G_{i,j})/N_i, \tag{19}$$

where $\sum_{i=1}^{|C|} N_i = N$. The following theorem shows that $\hat{\Phi}_{BCSS}$ is unbiased.

Theorem 5.3: $\mathbb{E}(\hat{\Phi}_{BCSS}) = \Phi_q(\mathcal{G})$.

The variance of $\hat{\Phi}_{BCSS}$ is given by

$$\mathrm{var}(\hat{\Phi}_{BCSS}) = \sum\nolimits_{i=1}^{|C|} (\pi_i^c)^2 \sigma_i / N_i, \tag{20}$$

Algorithm 3 *BCSS* ($\mathcal{G}$, N, q)

```
Input:  An uncertain graph $\mathcal{G} = (V, E, P)$, the sample size $N$,
        and a query $q$
Output: The BCSS estimator $\hat{\Phi}$

$\hat{\Phi} \leftarrow 0$;
Compute the cut set $C$ based on $q$, $\phi_q$, and $\mathcal{G}$;
for $i = 1$ to $|C|$ do
    Compute $\pi_i^c$ (Eq. (17)) and $\pi_i^{cd}$ (Eq. (21));
    $N_i \leftarrow [\pi_i^{cd} N]$, $t \leftarrow 0$;
    Let $X_i$ be the status vector of Stratum $i$;
    $\mathcal{G}_i \leftarrow$ simplify graph $\mathcal{G}$ based on $X_i$;
    for $j = 1$ to $N_i$ do
        Sampling $\mathcal{G}_i$ to generate a possible graph $G_j$;
        Compute $\phi_q(G_j)$, $t \leftarrow t + \phi_q(G_j)$;
    $t \leftarrow t/N_i$, $\hat{\Phi} \leftarrow \hat{\Phi} + \pi_i^c t$;
Calculate $\pi_0$ by Eq. (15) and $u_0$;
return $\hat{\Phi} + \pi_0^c u_0$;
```

where σ_i denotes the variance of the sample in Stratum i.

Sample allocation. Here we develop a new sample allocation strategy for *BCSS*. First, we define the conditional probability $\Pr[G \in \Omega_i | G \notin \Omega_0]$ as follows

$$\Pr[G \in \Omega_i | G \notin \Omega_0] = \pi_i^{cd} = \frac{p_i \prod_{j=1}^{i-1} (1 - p_j)}{1 - \prod_{j=1}^{r} (1 - p_j)}, \tag{21}$$

where $i = 1, \cdots, |C|$ and $\Pr[G \in \Omega_i | G \notin \Omega_0]$ denotes the probability of sampling a possible graph from Ω_i conditioning on that it is not in Ω_0. Second, our sample allocation strategy is given by $N_i = \pi_i^{cd} N$ for Stratum i. Based on this allocation strategy, we prove that the variance of $\hat{\Phi}_{BCSS}$ is no larger than that of the FS estimator.

Theorem 5.4: $\mathrm{var}(\hat{\Phi}_{BCSS}) \le \mathrm{var}(\hat{\Phi}_{FS})$.

Similar to *BSS-II*, we can also apply the graph simplification technique into the *BCSS* algorithm. The detailed description of the algorithm is shown in Algorithm 3. Also, by a similar assumption as made in *BSS-II*, the time complexity of Algorithm 3 is $O(N(m + M) + T)$, where computing the cut set takes $O(T)$ time. It is worth mentioning that in many applications the cut set can be easily calculated, and the time complexity $O(T)$ can be dominated by $O(m)$. As a result, the time complexity of Algorithm 3 is the same as that of *NMC*.

5.3 The *RCSS* estimator

Similar to *RSS-I* and *RSS-II*, the *BCSS* estimator can also be recursively applied in each stratum. Based on this idea, we propose a recursive cut-set based stratified sampling estimator, named *RCSS* estimator. The *RCSS* algorithm is outlined in Algorithm 4. Likewise, the *RCSS* estimator is unbiased and its variance is no larger than that of *BCSS*. Also, we can show that the time complexity of Algorithm 4 is $O(N(M + m + T))$ based on the same assumption as made in *RSS-II*. In many real-world applications, $O(T)$ can be dominated by $O(m)$, thus in these cases the time complexity of *RCSS* is the same as that of *NMC*. The detailed description of Algorithm 4 and its complexity analysis are given in the supplementary document of this paper.

5.4 Discussion

Here we give a brief discussion of all the proposed estimators. First, recall that the stratification method of *BCSS*

Algorithm 4 RCSS($\mathcal{G}$, N, q, ρ, τ)

```
Input:  An uncertain graph $\mathcal{G} = (V, E, P)$, the sample size $N$
        a query $q$, and parameters $\rho$ and $\tau$
Output: The RCSS estimator $\hat{\Phi}$

$\hat{\Phi} \leftarrow 0$;
Compute the cut set $C$ based on $q$, $\phi_q$, and $\mathcal{G}$;
if $N < \tau$ or $|E| < \rho$ or $|C| == 0$ then
    for $j = 1$ to $N$ do
        Sampling $\mathcal{G}$ to generate a possible graph $G_j$;
        Compute $\phi_q(G_j)$, $\hat{\Phi} \leftarrow \hat{\Phi} + \phi_q(G_j)$;
    return $\hat{\Phi}/N$;
else
    Compute $\pi_0^c$ and $u_0$ based on the cut set $C$;
    for $i = 1$ to $|C|$ do
        Let $X_i$ be the status vector of set $C$ in Stratum $i$;
        $\mathcal{G}_i \leftarrow$ simplify graph $\mathcal{G}$ based on $X_i$;
        Compute $\pi_i^c$ (Eq. (17)) and $\pi_i^{cd}$ (Eq. (21));
        $N_i \leftarrow [\pi_i^{cd} N]$;
        $\mu_i \leftarrow$RCSS($\mathcal{G}_i$, $N_i$, $q$, $\rho$, $\tau$);
        $\hat{\Phi} \leftarrow \hat{\Phi}$ + $\pi_i^c \mu_i$;
    return $\hat{\Phi}$ + $\pi_0^c \mu_0$;
```

is very similar to that of *BSS-II*. The differences between these two methods are: (1) the stratification method of *BCSS* is based on the cut set while the stratification of *BSS-II* is based on any selected r edges, and (2) unlike *BSS-II*, there is no Stratum 0 in *BCSS*. Moreover, we find that if $r = |C|$, the variance of *BCSS* is no larger than that of *BSS-II*.

Theorem 5.5: *If* $r = |C|$, $\mathrm{var}(\hat{\Phi}_{BCSS}) \leq \mathrm{var}(\hat{\Phi}_{BSSII})$.

Theorem 5.5 implies that the *BCSS* estimator reduces the variance of *BSS-II* under the condition of $|C| = r$. Note that it is very hard to compare the variances between *BCSS* and *BSS-II* for $|C| \neq r$ because in this case the strata of these two methods are totally different. By the same reason, it is also very hard to compare the variances between *BCSS* and *BSS-I* and compare the variances among *RCSS*, *RSS-I*, and *RSS-II* as well. In addition, it should be noted that the class-I and class-II estimators are very general which are independent of the graph structure and the property of the query evaluation function. However, both the *BCSS* and *RCSS* estimators exploit the graph structure and the cut-set property of the query evaluation function, thus these estimators can be deemed as *data-driven* methods where the stratification methods are driven by the cut set. In our experiments, we will show that such data-driven methods significantly outperform the class-I and class-II estimators.

6 Applications

In this section, we apply the proposed algorithms with graph simplification technique to three different uncertain graph query problems, which are the influence function evaluation problem [8], [11], the expected-reliable distance query problem [2], as well as the distance-constraint reachability problem [3]. For the two former problems, the original implementations of our algorithms (without integrating the graph simplification technique) can be found in the preliminary version of this paper [13]. It should be noted that developing an efficient graph simplification technique for the recursive stratified sampling algorithms is a nontrivial task. This is because in our algorithms, we need to invoke the graph simplification procedure in each stratum, thus the total time complexity of the graph simplification procedure may be very high. Below, we show how to tackle this challenge for three different uncertain graph query problems.

6.1 Influence function evaluation

Given an uncertain graph $\mathcal{G}$ and a seed set A, the influence function evaluation (IFE) problem is to compute the expected number of nodes in $\mathcal{G}$ that are reachable from the seed set A. This problem plays a crucial role in influence maximization problem in social networks [8], [11]. To simplify our presentation, we consider a special case that the seed set only contains one node, i.e., $|A| = 1$. For the general case ($|A| > 1$), the problem can be easily converted to the problem with only one seed node. This is because we can add a virtual node q and $|A|$ edges from q to each node in A with probability 1, then the IFE problem with seed set A is equivalent to the problem with seed node q. Clearly, such a problem is an instance of the expectation query evaluation problem. The query is a seed node, and the query evaluation function on a possible graph G, i.e., $\phi_q(G)$, denotes the number of nodes that are reachable from the seed node q in G. The query evaluation function can be calculated by the *BFS* algorithm. Beyond the IFE problem, here we also introduce the threshold IFE problem. In particular, given an uncertain graph $\mathcal{G}$, a seed node q, a threshold δ, the threshold IFE problem aims at estimating the probability of $\phi_q(G) \geq \delta$ for a possible graph G. In the following, we focus on the IFE problem. The algorithms can be easily generalized to the threshold IFE problem because we only need to replace $\phi_q(G)$ by $\mathrm{I}(\phi_q(G) \geq \delta)$.

Graph simplification for class-I estimators. For the class-I estimators (*BSS-I* and *RSS-I*), a straightforward graph simplification algorithm is to invoke a *BFS* procedure that starts from the seed node q to traverse the uncertain graph $\mathcal{G}$ (ignore the probabilities of the edges), and all the nodes that cannot be reachable from q can be pruned. However, this method is rather inefficient. This is because in *BSS-I* and *RSS-I*, we need to invoke such a algorithm in each stratum for graph simplification (see line 5 in Algorithm 1 and line 12 in Algorithm 2). Since there will be 2^r strata in each stratification, we has to invoke such a *BFS* algorithm 2^r times in each stratification, which takes $O(2^r m)$ time complexity in the worst case. Below, we present a new algorithm that only invokes the *BFS* procedure $r+1$ times in each stratification, which is clearly much more efficient than the straightforward algorithm.

Specifically, in the stratification, we only invoke the *BFS* procedure for the first $r+1$ strata, i.e., the first $r+1$ rows in Table 1. Then, for each Stratum i ($i = 1, \cdots, r+1$), we can obtain a simplified uncertain graph $\mathcal{G}_i$ after invoking a *BFS*. For the other strata (i.e., Stratum i for $i = r+2, \cdots, 2^r$), the corresponding simplified uncertain graphs can be obtained by merging some of $\mathcal{G}_i$ ($i = 1, \cdots, r+1$) without invoking *BFS*. The key idea our algorithm is based on Theorem 6.1. Before giving the theorem, we first show a useful lemma below.

Lemma 6.1: *Let X_i be the status vector of Stratum i (Table 1). Then, for $i = r+2, \cdots, 2^r$, each X_i can be represented by the element-wise addition among some of* *the first $r+1$ status vectors (X_i with $i = 1, \cdots, r+1$), i.e., $X_i = \sum_{j \in S, S \subseteq \{1,\cdots,r+1\}} X_j$.* □

For example, suppose that $r = 3$. The first $r + 1$ status vectors are $(0, 0, 0)$, $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$. Then, for the status vector $(1, 1, 0)$, it can be represented by the element-wise addition between the status vectors $(1, 0, 0)$ and $(0, 1, 0)$.

Theorem 6.1: *For each Stratum i ($i = r + 2, \cdots, 2^r$), if $X_i = \sum_{j \in S, S \subseteq \{1,\cdots,r+1\}} X_j$ (element-wise summation), then the simplified uncertain graph G_i can be obtained by $G_i = \bigcup_{j \in S, S \subseteq \{1,\cdots,r+1\}} G_j$.*

With Theorem 6.1, the total time complexity of the graph simplification procedure in *BSS-I* is bounded by $O(rm)$. Since r must be a very small constant in the *BSS-I* algorithm (e.g., $r = 5$), we have $O(rm) = O(m)$. For the *RSS-I* algorithm, we only invoke the graph simplification procedure in the first stratification to achieve the $O(m)$ time complexity. For the other stratifications in *RSS-I*, we do not perform graph simplification any more. In the experiments, we will show that the *RSS-I* algorithm with such a *lightweight* graph simplification method not only boosts the accuracy of the estimator, but it can also reduce the running time of the algorithm (because we only need to sample a relatively small uncertain graph after graph simplification).

Graph simplification for class-II estimators. For the class-II estimators, a similar straightforward algorithm for graph simplification is to perform *BFS* starting from q to find all the irrelevant nodes, which cannot be reachable from q. However, for a stratification in *BSS-II* and *RSS-II*, the straightforward algorithm invokes the *BFS* procedure $r + 1$ times, which takes $O(rm)$ time in total. Unlike *BSS-I* and *RSS-I*, r could be a relatively large constant in *BSS-II* and *RSS-II* (e.g., $r = 50$), thus the straightforward algorithm is costly. Below, we develop a much more efficient graph simplification algorithm, called progressive *BFS*, which progressively traverses the whole uncertain graph by a *BFS* procedure.

The rough idea of the progressive *BFS* algorithm is as follows. First, the algorithm performs a *BFS* to simplify the residual uncertain graph denoted by $\bar{\mathcal{G}}_0$ after removing all the selected r edges (corresponding to simplify the uncertain graph in Stratum 0). Then, the algorithm iteratively executes r times. In iteration i for $i = 1, \cdots, r$, the algorithm adds back one removed edge (one of the selected r edges) into $\bar{\mathcal{G}}_{i-1}$ which results in a new residual graph $\bar{\mathcal{G}}_i$, and then performs a pruned *BFS* on $\bar{\mathcal{G}}_i$, in which the pruned *BFS* does not traverse the nodes that have already been visited by the previous *BFS* procedures (corresponding to simplify the uncertain graph in Stratum $r - i + 1$).

More specifically, the progressive *BFS* algorithm first simplifies the uncertain graph in Stratum 0 (Table 2). In particular, the algorithm performs a traditional *BFS* from q on the residual uncertain graph (ignore probabilities of the edges) after removing all the selected r edges. We use a bitmap with $O(n)$ bits to record the nodes whether they are visited by the *BFS* or not. Then, the algorithm simplifies the residual uncertain graphs from Stratum r to Stratum 1. To better describe our algorithm, we refer to the edge with status ‘1’ in Stratum i ($i = 1, \cdots, r$) as the active edge (Recall that there is only one active edge in each Stratum). When the algorithm simplifies the residual uncertain graph in Stratum i ($i = 1, \cdots, r$), the algorithm first traverses the active edge in Stratum i, and then performs a pruned *BFS* algorithm to traverse the residual uncertain graph (the graph after removing the failed edges in Stratum i), where the pruned *BFS* does not traverse the nodes that are visited by the previous *BFS* procedures. During this procedure, the algorithm need to update the bitmap to record the nodes that are newly visited by the pruned *BFS*. When the pruned *BFS* terminates, the simplified uncertain graph for Stratum i is the graph that includes all the visited nodes and edges so far.

Let $\mathcal{G}_i$ be the simplified uncertain graph for Stratum i ($i = 0, \cdots, r$) by the progressive *BFS* algorithm. Then, we have the following theorem.

Theorem 6.2: *For $i = 2, \cdots, r$, we have $\mathcal{G}_i \subseteq \mathcal{G}_{i-1}$, and $\mathcal{G}_0 \subseteq \mathcal{G}_r$, where $\subseteq$ denotes the inclusion relationship.*

Based on Theorem 6.2, we can show the correctness of the progressive *BFS* algorithm.

Theorem 6.3: *The progressive BFS algorithm is correct.*

For the time complexity, it is easy to show that each edge in the uncertain graph is traversed at most once by the whole progressive *BFS* procedure, thus the total time overhead of the algorithm is $O(m)$.

Below, we show how to generalize the progressive *BFS* algorithm for *RSS-II*. Recall that the *RSS-II* algorithm will produce a recursion tree, where each tree node denotes a stratification. It should be noted that each tree node also corresponds to a stratum, because in that tree node the algorithm will recursively partition a stratum. In addition, for each stratum, we have a residual uncertain graph after removing all the failed edges in that stratum. A naive generalization of the progressive *BFS* algorithm is to perform a progressive *BFS* in each stratification, which is costly because there may be too many tree nodes (stratifications). Fortunately, we find that we can only perform a progressive *BFS* in each level of the recursion tree to achieve the same goal. Specifically, we assume that in each tree node, the children of that node from left to right denote the Stratum 0, Stratum r, $\cdots$, Stratum 1, respectively as illustrated in Fig. 2 (a three-level recursion tree). In each level, we start *BFS* for the leftmost tree node to simplify the uncertain graph corresponding to that tree node, and then perform the pruned *BFS* for the sibling nodes following a “left to right” order (see Fig. 2). It can be shown that the simplified uncertain graph obtained in a tree node is the same as the union of the simplified graphs obtained in its children (using a similar analysis of the progressive *BFS* algorithm for *BSS-II*). As a result, the union of the simplified graphs obtained in all the tree nodes in a level must be the same as the simplified graph obtained in the root node. Since we make use of the pruned *BFS* to simplify the graphs, thus all the edges are visited at most once by the whole procedure in each level of the recursion tree. Therefore, the total time complexity of the graph simplification procedure is $O(\bar{d}m)$, where $\bar{d}$ denotes the height of the recursion tree. For *RSS-II*, r is a relatively large value (e.g. $r = 50$), thus the height of the recursion tree $\bar{d}$ is $O(\log_r^N)$ which is very small (e.g., when $r = 50$ and $N = 10,000$, $\bar{d} < 3$). Thus, the time

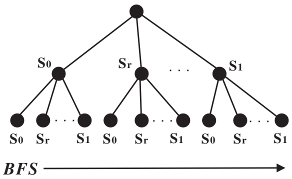


Fig. 2. Illustration of the idea of the graph simplification procedure for *RSS-II*. S_i denotes Stratum i.

complexity of the whole graph simplification procedure for *RSS-II* can be bounded by $O(m)$.

Graph simplification for cut-set based estimators. For the cut-set based estimators, we define the cut set as the set of all the outgoing edges of the seed node q. This is because if all the outgoing edges of q are failed, then no node can be reachable from q, and thus $\phi_q(G)$ is a constant 0, satisfying the definition of cut set. Based on the cut set, we can devise a very similar graph simplification procedure as the class-II estimators for the cut-set based estimators. The only difference is that we do not need to simplify the uncertain graph in Stratum 0 (see Tables 2 and 3). Also, by a similar analysis as the class-II estimators, the time complexity of the whole graph simplification procedure for the cut-set based estimators is $O(m)$.

6.2 Expected-reliable distance query

The expected-reliable distance (ERD) is an important measure for k-nearest neighbor query on uncertain graphs [2]. Given an uncertain graph $\mathcal{G}$ and two query nodes s and t, the expected-reliable distance query is to estimate

$$\Phi_{s,t}(\mathcal{G}) = \sum\nolimits_{G\in\Omega\setminus\Omega_\infty} \Pr[G]\phi_{s,t}(G)/(1-\Pr[G\in\Omega_\infty]), \tag{22}$$

where $\phi_{s,t}(G)$ is the query evaluation function representing the length of the shortest path from s to t (the distance from s to t), and Ω_∞ denotes the probability space in which s cannot reach t, i.e., $\phi_{s,t}(G) = \infty$ for $G \in \Omega_\infty$. Besides the ERD query problem, we also study its threshold counterpart. More specifically, given an uncertain graph $\mathcal{G}$, two query nodes s and t, and a threshold δ, the threshold ERD query is to estimate the probability of $\phi_{s,t}(G) \le \delta$ for a possible graph G. Since the algorithm for the threshold ERD query is very similar to the algorithm for the ERD query, we focus on the ERD query. Note that we can easily apply the proposed algorithms to this problem. The main technical challenge is how to develop an efficient graph simplification technique for our algorithms.

Our graph simplification is based on the following observation. For a query (s,t), we let R_s be a set of nodes that can be reachable from s, and R_t be a set of nodes that can reach t in the uncertain graph $\mathcal{G}$ (ignore probabilities). Then, we can prune all the nodes that are not in $R_s \bigcap R_t$, because those nodes cannot appear in the shortest path between s and t. Recall that for the influential function evaluation (IFE) problem, we prune the nodes that cannot be reachable from the query node q, and record the set of all reachable nodes R_q. Based on this, we can apply the same graph simplification algorithm as used for the IFE problem to the ERD query problem. Specifically, for the class-I estimators, in Stratum i ($i = 1, \cdots, 2^r$), we let R_s^i be the set of reachable nodes from s and R_t^i be the set of nodes that can reach t. Then, we perform the *BFS* procedures $r+1$ times for computing R_s^i in the first $r+1$ strata respectively, and then we use the same method as used in Section 6.1 to compute the other R_s^i in Stratum $r+2, \cdots$, Stratum 2^r respectively. Similarly, we can used the same method to compute R_t^i in each stratum. The simplified uncertain graph in Stratum i can be obtained by $R_s^i \bigcap R_t^i$. The whole procedure takes $O(rm) = O(m)$ time complexity, as r is very small constant. Likewise, for the class-II and cut-set based estimators, we perform the progressive *BFS* algorithm to compute R_s^i and R_t^i respectively, and then take $R_s^i \bigcap R_t^i$ to determine the simplified uncertain graph in Stratum i. It is easy to figure out that the time complexity of the whole procedure is also $O(m)$.

6.3 Distance-constraint reachability query

The distance-constraint reachability (DCR) query [3] is an instance of the threshold query evaluation problem, where the query is given by a source node s, a target node t, and the distance threshold δ. The goal of the query is to compute the probability that the distance between s and t is less than or equal to the threshold d. To estimate such a probability, Jin et al. propose an efficient algorithm in [3], which is a special case of our *RSS-I* algorithm when $r = 1$ (see Section 3.2). Clearly, it is very easy to apply the proposed algorithms to the DCR query problem. Before using our algorithms for evaluating the query, we can make use of the pruning technique proposed in [3] to simplify the uncertain graph. Specifically, the pruning technique [3] is based on the following observation. Let $d(s,v)$ denotes the shortest path distance from s to v, and $d(v,t)$ denotes the shortest path distance from v to t in the uncertain graph (ignore the probabilities). Then, for the nodes that are not in $\tilde{V} = \{v \in V | d(s,v) + d(v,t) \le \delta\}$ can be pruned. We can use two Dijkstra procedures (within diameter δ) to compute $\tilde{V}$, which takes nearly linear time complexity [3]. Then, we execute our algorithms on this simplified uncertain graph. The main challenge is how to develop an efficient graph simplification procedure for our algorithms.

For our algorithms, a straightforward graph simplification procedure is to use the above pruning technique to find the simplified uncertain graph. However, this is fairly inefficient, because we need to invoke the above pruning technique in each stratification (there may be too many stratifications in *RSS-I*, *RSS-II* and *RCSS*). Moreover, it is not clear how to reduce the total computational cost of the whole graph simplification procedure for the DCR query into $O(m)$ time (like our simplification techniques developed for IFE problem and ERD query). To develop an efficient graph simplification technique for the DCR query, we resort to the graph simplification procedure used for the ERD query. This is because the nodes that are not reachable from s or cannot reach t can be definitely pruned for both DCR and ERD queries. Therefore, in our implementation, we apply the same graph simplification procedure that is used for the ERD query to the DCR query, which takes $O(m)$ time in total.

7 Experiments

In this section, we evaluate the accuracy and efficiency of the proposed algorithms for the influence function evaluation (IFE), the expected-reliable distance (ERD) query, and the distance-constraint reachability (DCR) query. We also study the performance of our algorithms for the threshold

TABLE 4
Summary of the datasets

Name	Nodes	Edges	Ref.
ER	5,000	50,616	[3]
Facebook	1,899	20,296	[15]
Condmat	16,264	95,188	[16]
DBLP	78,648	376,515	[14]
WikiCon	118,100	2,917,785	Website
EastUSA	3,598,623	8,778,114	Website

IFE and ERD queries as well. Due to space limit, the results for the threshold IFE and ERD queries are shown in the supplementary document of this paper.

Datasets. We use one synthetic dataset and five real-world datasets in our experiments. We apply the same parameters used in [3] to generate the synthetic dataset. In particular, we first generate an Erdos-Renyi (ER) random graph with 5,000 vertices and 50,616 edges. Then, for each edge, we generate a probability according to a [0,1] uniform distribution. The five real-world datasets are as follows. (1) Facebook dataset: this dataset originates from a Facebook social network for students at University of California, Irvine. It contains the users who sent or received at least one message. We collect this dataset from (toreopsahl.com/datasets). The dataset is a weighted graph, and the weight of each edge denotes the number of messages passing over the edge. (2) Condmat dataset: this dataset is a weighted collaboration network, where the weight of an edge represents the number of co-authored papers between two collaborators. We download this dataset from (www-personal.umich.edu/∼mejn/netdata). (3) DBLP dataset: this dataset is also a weighted collaboration network, where the weight of the edge signifies the number of co-authored papers. This dataset is provided by the authors in [14]. (4) WikiCon dataset: this dataset is a conflict network where the weights of the edges represent the conflicts between users of the English Wikipedia. We download this dataset from (konect.uni-koblenz.de/networks/wikiconflict). (5) EastUSA dataset: this is a road network dataset of eastern USA, where the weights of the edges represent the transit time between two locations. We download this dataset from (http://www.dis.uniroma1.it/challenge9). Table 4 summarizes the detailed information of our datasets. To obtain the uncertain networks, for each real-world dataset, we generate the probabilities using the method proposed in [2], [3]. Specifically, to generate the probability of an edge, we apply an exponential cumulative distribution function (CDF) with mean 2 to the weight of that edge.

Different estimators. We compare eleven different estimators. The first two estimators are served as the baselines, and the last nine estimators are our proposed recursive estimators. The different estimators are summarized as follows. (1) NMC, which is the naive Monte-Carlo estimator. (2) RSSIR1, which is a special *RSS-I* estimator with random (*RM*) edge-selection strategy and with parameter $r = 1$. This estimator is presented in [3] for computing distance-constraint reachability on uncertain graphs. (3) *RSS-I*, *RSS-II*, and *RCSS* which are the proposed recursive estimators without integrating the graph simplification technique. (4) *BSS-I**, *BSS-II**, *BCSS**, *RSS-I**, *RSS-II**, and *RCSS**, which are the proposed recursive estimators with graph simplification technique. For all the proposed estimators, we use the *BFS* edge-selection strategy to pick the r edges for stratification, because it is more effective than the *RM* edge-selection strategy as shown in [13]. It should be noted that we do not compare the proposed basic estimators without graph simplification as they are shown to be less effective than the proposed recursive estimators [13].

Evaluation metric. Two metrics are used to evaluate the performance of different estimators: average query time and relative variance. The average query time evaluates the efficiency of the estimators. The relative variance is leveraged to evaluate the accuracy of the estimators. Let σ_{NMC} be the variance of the *NMC* estimator. We calculate the relative variance of an estimator $\hat{\Phi}$ by $\sigma_{\hat{\Phi}}/\sigma_{NMC}$. Since computing the exact variance of an estimator is intractable, we resort to an unbiased estimator of the variance. Similar evaluation metric has been used in [3] for the distance-constraint reachability problem. Specifically, to get the unbiased estimator of the variance, we run each estimator $(\hat{\Phi}_i(\mathcal{G}))$ 500 times. Then, the unbiased estimator of the variance is obtained by $\sum_{i=1}^{500}(\hat{\Phi}_i(\mathcal{G}) - \bar{\Phi}_i(\mathcal{G}))^2/499$, where $\bar{\Phi}_i(\mathcal{G})$ denotes the mean of $\hat{\Phi}_i(\mathcal{G})$ $(i = 1, \cdots, 500)$.

Parameter settings and experimental environment. For all estimators, we set the sample size $N = 1000$. For the class-I and class-II estimators, we set $r = 5$ and $r = 50$ respectively, because under this setting these estimators perform very well. We will also study the effect of r in these estimators in the experiments. For the threshold parameter τ in *RSS-I* (*RSS-I**) and *RSS-II* (*RSS-II**), we set $\tau = 10$, and for the threshold parameters in *RCSS* (*RCSS**) we set $\tau = 10$ and $\rho = 10$. All the experiments are conducted on a Scientific Linux 6.0 workstation with 2xQuad-Core Intel(R) 2.66 GHz CPU, and 32G memory. All algorithms are implemented in C++.

7.1 Experimental results

In all the experiments, we randomly generate 1000 queries for three different queries: IFE, EDR, and DCR. The reported results are the average result over all the queries.

Exp-1: Accuracy of different estimators. In this experiment, we evaluate the accuracy of eleven different algorithms. The results of different estimators for IFE, EDR, and DCR queries are reported in Tables 5, 6, and 7, respectively. As can be seen, both *RCSS** and *RCSS* consistently outperform the other algorithms over all the datasets for all the IFE, EDR, and DCR queries. This is because these two algorithms exploit the graph structural information (the cut set) for stratification which significantly reduce the variances of the estimators by cutting an *irrelevant* stratum in each stratification (see Table 3). Also, we can see that the other proposed recursive estimators perform very well for all the IFE, EDR, and DCR queries, which drastically reduce the relative variance over the state-of-the-art estimator RSSIR1. For example, in the Condmat dataset in Table 5, *RSS-I*, *RSS-II*, *RCSS*, *RSS-I**, *RSS-II**, and *RCSS** cut the relative variance over RSSIR1 by 403.5%, 375.0%, 580.3%, 523.0%, 576.1%, and 597.7% respectively. Moreover, we can see that the recursive estimators with graph simplification technique consistently outperform the corresponding recursive estimators without graph simplification. The reasons are as follow. On the one hand, the graph simplification procedure reduces the

TABLE 5
Influence function evaluation: Comparison of relative variance (RV) of different estimators

RV	NMC	RSSIR1	BSS-I*	BSS-II*	BCSS*	RSSI	RSSII	RCSS	RSSI*	RSSII*	RCSS*
ER	1.000	0.672	0.726	0.732	0.216	0.209	0.206	0.153	0.185	0.152	**0.149**
Facebook	1.000	0.559	0.643	0.651	0.321	0.257	0.240	0.168	0.204	0.169	**0.162**
Condmat	1.000	0.795	0.812	0.810	0.239	0.197	0.212	0.137	0.152	0.138	**0.133**
DBLP	1.000	0.538	0.601	0.589	0.203	0.192	0.182	0.125	0.169	0.129	**0.123**
WikiCon	1.000	0.603	0.631	0.633	0.216	0.198	0.193	0.131	0.172	0.138	**0.130**
EastUSA	1.000	0.682	0.691	0.685	0.218	0.201	0.195	0.135	0.181	0.139	**0.132**

TABLE 6
Expected-reliable distance query: Comparison of relative variance (RV) of different estimators

RV	NMC	RSSIR1	BSS-I*	BSS-II*	BCSS*	RSSI	RSSII	RCSS	RSSI*	RSSII*	RCSS*
ER	1.000	0.772	0.882	0.891	0.875	0.679	0.675	0.398	0.601	0.425	**0.395**
Facebook	1.000	0.759	0.874	0.880	0.871	0.648	0.650	0.356	0.562	0.398	**0.353**
Condmat	1.000	0.815	0.832	0.833	0.829	0.695	0.691	0.350	0.568	0.361	**0.348**
DBLP	1.000	0.756	0.852	0.853	0.846	0.683	0.680	0.515	0.631	0.520	**0.513**
WikiCon	1.000	0.789	0.861	0.859	0.844	0.690	0.685	0.526	0.632	0.531	**0.526**
EastUSA	1.000	0.792	0.859	0.852	0.834	0.693	0.686	0.531	0.629	0.534	**0.529**

TABLE 7
Distance-constraint reachability query: Comparison of relative variance (RV) of different estimators

RV	NMC	RSSIR1	BSS-I*	BSS-II*	BCSS*	RSSI	RSSII	RCSS	RSSI*	RSSII*	RCSS*
ER	1.000	0.235	0.685	0.691	0.521	0.220	0.219	0.182	0.218	0.193	**0.180**
Facebook	1.000	0.231	0.643	0.646	0.516	0.231	0.236	0.185	0.226	0.189	**0.181**
Condmat	1.000	0.385	0.678	0.675	0.531	0.325	0.319	0.225	0.312	0.228	**0.224**
DBLP	1.000	0.432	0.720	0.713	0.562	0.331	0.328	0.236	0.319	0.238	**0.233**
WikiCon	1.000	0.512	0.732	0.724	0.570	0.355	0.351	0.218	0.347	0.225	**0.216**
EastUSA	1.000	0.573	0.715	0.709	0.568	0.328	0.324	0.253	0.320	0.255	**0.250**

uncertainty of the graph in each stratification, thus may cut the variance. On the other hand, the graph simplification procedure prunes many irrelevant edges, thus can avoid the algorithms to select the irrelevant edges for stratification. In addition, we find that *RSS-II** performs much better than *RSS-I**, and its performance is comparable to that of *RCSS**. The reason could be that *RSS-II** makes use of the progressive *BFS* algorithm for graph simplification in each stratification, while *RSS-I** only invokes the graph simplification procedure in the first stratification to achieve the $O(m)$ time cost. Another additional observation is that the proposed basic estimators with graph simplification also perform well for all the queries, which are significantly better the *NMC* estimator and slightly worse than RSSIR1 in general. These results confirm the theoretical analysis in the previous sections.

Exp-2: Efficiency of different estimators. We report the average query time of eleven various algorithms for IFE, EDR, and DCR queries in Tables 8, 9, and 10, respectively. From Tables 8, 9, and 10, we can see that the efficiency of the algorithms without graph simplification are comparable to that of *NMC*, and the algorithms with graph simplification significantly shorten the average query time over the algorithms without graph simplification. These results are consistent with the complexity analysis shown in the previous sections. Notably, *RSS-II** is faster than *RCSS** (because *RCSS** needs to compute the cut set in each stratification), and both of them nearly halve the running time of *RSS-II* and *RCSS*. In addition, it should be noted that the RSSIR1 algorithm is also very fast for the DCR query. This is because such an algorithm is originally tailored for the DCR query [3] which can be integrated with an effective pruning technique as discussed in Section 6.3. However, for the IFE and EDR queries, it is not clear how to devise an effective pruning technique for RSSIR1.

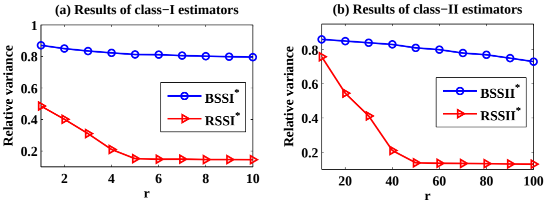


Fig. 3. Effect of parameter r

Exp-3: Effect of parameter r. We study the effectiveness of the parameter r in the proposed class-I and class-II estimators (with graph simplification) for the IFE query using the Condmat dataset. Similar results can be observed for both ERD and DCR queries and using other datasets. Fig. 3(a) and Fig. 3(b) show the relative variances of the class-I and class-II estimators w.r.t. various r. As can be seen in Fig. 3(a), the relative variances of *BSS-I** and *RSS-I** decrease with increasing r when $r \leq 5$, whereas if $r > 5$, the curves tend to be smooth. The reason could be that if r is large, then there will be a large number of strata generated. Since the sample size is limited, the algorithm will draw a very small number of samples in some strata, which may slightly increase the variance of the estimator in those strata. Based on our observation, $r = 5$ is a very good choice for the class-I estimators, which is also used in the previous experiments. Similarly, for the class-II estimators, we find that the relative variances of *BSS-II** and *RSS-II** decrease as r increases when $r \leq 50$, and when $r \geq 50$ the curves tend to be smooth. Therefore, $r = 50$ is a good choice for the class-II estimators, which is also used in our previous experiments.

Exp-4: Effect of sample size. As shown in the previous experiments, *RSS-II** and *RCSS** outperform the other estimators (in terms of both accuracy and average query time). Here we study how the sample size affects the

TABLE 8
Influence function evaluation: Comparison of average query time of different estimators (seconds)

RV	NMC	RSSIR1	BSS-I*	BSS-II*	BCSS*	RSSI	RSSII	RCSS	RSSI*	RSSII*	RCSS*
ER	0.359	0.356	0.353	0.352	0.359	0.378	0.385	0.396	**0.345**	0.349	0.351
Facebook	0.201	0.201	0.202	0.203	0.205	0.201	0.204	0.235	0.198	**0.191**	0.203
Condmat	1.297	1.296	1.258	1.252	1.295	1.241	1.228	1.306	1.220	**1.101**	1.285
DBLP	8.582	8.654	7.653	7.589	9.619	8.593	8.684	9.628	7.782	**6.382**	6.586
WikiCon	72.32	74.25	56.87	58.32	57.25	84.19	82.68	86.31	57.23	**40.21**	45.68
EastUSA	2018	2068	1485	1461	1503	2090	2088	2095	1493	**1021**	1109

TABLE 9
Expected-reliable distance query: Comparison of average query time of different estimators (seconds)

RV	NMC	RSSIR1	BSS-I*	BSS-II*	BCSS*	RSSI	RSSII	RCSS	RSSI*	RSSII*	RCSS*
ER	0.405	0.422	0.403	**0.401**	0.415	0.431	0.427	0.453	0.430	0.423	0.450
Facebook	0.210	0.231	0.208	0.211	0.212	0.216	0.236	0.241	0.208	**0.206**	0.221
Condmat	1.383	1.387	1.375	1.369	1.381	1.401	1.408	1.410	1.371	**1.312**	1.339
DBLP	11.33	11.41	8.631	8.525	8.598	11.46	11.43	11.48	8.531	**6.725**	8.319
WikiCon	93.21	95.26	75.37	72.25	75.31	99.81	95.63	105.3	76.35	**50.23**	55.69
EastUSA	2784	2843	2016	2110	2125	2882	2892	2889	2025	**1098**	1169

TABLE 10
Distance-constraint reachability query: Comparison of average query time of different estimators (seconds)

RV	NMC	RSSIR1	BSS-I*	BSS-II*	BCSS*	RSSI	RSSII	RCSS	RSSI*	RSSII*	RCSS*
ER	0.313	0.302	0.310	0.311	0.315	0.314	0.314	0.329	0.301	**0.290**	0.305
Facebook	0.200	0.181	0.189	0.185	0.191	0.210	0.206	0.221	0.182	**0.178**	0.193
Condmat	1.253	1.216	1.255	1.259	1.267	1.320	1.321	1.432	1.218	**1.202**	1.249
DBLP	7.521	6.541	7.231	7.426	7.502	8.043	8.025	8.126	6.631	**6.125**	6.389
WikiCon	63.81	45.20	53.32	54.87	55.26	63.89	64.78	70.21	50.87	**35.31**	40.69
EastUSA	635.2	432.3	569.5	563.8	581.8	641.5	639.8	653.1	578.6	**312.3**	393.5

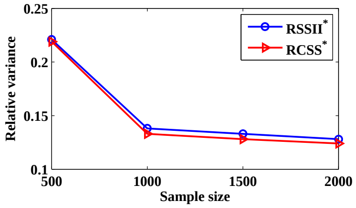


Fig. 4. Relative variance vs. sample size

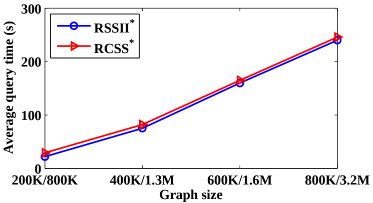


Fig. 5. Scalability testing

accuracy of these estimators for the IFE query using the Condmat dataset. Similar results can also be observed for the other queries and in other datasets. Fig. 4 shows the relative variances of the estimators under different sample sizes. As can be observed, the curves of *RSS-II** and *RCSS** are relatively smooth when the sample size is no less than 1000, indicating that the relative variances of these estimators are relatively robust w.r.t. the sample size.

Exp-5: Scalability testing. In this experiment, we study the scalability of our best estimators *RSS-II** and *RCSS**. To this end, we generate four large synthetic uncertain graphs with the number of nodes ranging from 200,000 (200k) to 800,000 and the number of edges ranging from 800,000 to 3,200,000 (3.2m). Also, for each estimator, we set the same parameter setting as our previous experiments. Fig. 5 depicts the average query time of the estimators for the IFE query. Similar results can also be observed for the ERD and DCR queries. In Fig. 5, the two numbers in the horizontal axis (eg. 200k/800k) denote the number of nodes and the number of edges respectively. From Fig. 5, we find that the average query time increases as the graph size increases. Furthermore, the two estimators exhibit a linear growth w.r.t. the graph size. These results demonstrate that our best estimators scale linearly w.r.t. the graph size, which are consistent with the time complexity of the estimators.

8 RELATED WORK

Uncertain graph management and mining has been attracted much attention due to the increasing applications in biological database [17], communication networks [7], and influence networks [8]. Notable studies on uncertain graph management and mining include reliable subgraph search [18], [19], [4], reachability computation [3], [20], frequent subgraph mining [1], [21], k-nearest neighbor search [2], reliable clustering [22], as well as subgraph pattern matching [23]. Generally, all the mentioned uncertain graph problems are known to be #P-complete, thus finding the exact solution is impossible in large uncertain graphs unless P=#P. Therefore, most existing studies, such as [2] and [4], are based on the *NMC* estimator. Generally, *NMC* leads to a large variance, thus reducing the accuracy of the algorithms. To reduce the variance, in our preliminary work [13], we develop several accurate *RSS* (recursive stratified sampling) estimators without sacrificing efficiency for the query evaluation problems on uncertain graphs. In the present work, we substantially extend our previous work by integrating an elegant graph simplification technique into the *RSS* estimators. Our graph simplification technique not only significantly improves the accuracy of the estimators, but it also reduce the running time of the estimators. We apply the proposed techniques to three different uncertain graph query evaluation problems, and new experimental studies are also conducted as well.

In addition, it is worth mentioning that Parchas et al. [24] proposed an interesting algorithm to find the best possible graph to represent the original uncertain graph, and then use such a representative possible graph for graph query and mining tasks. However, their method aims at preserving the expected vertex degree of the uncertain graph, it is not clear whether it preserves the other graph structural properties, thus may not be used for general uncertain graph query evaluation problems.

9 Conclusion

In this paper, we propose several efficient and accurate estimators based on recursive stratified sampling to solving two types of uncertain graph query evaluation problems. We show that all of proposed estimators are unbiased and their variances are smaller than that of the state-of-the-art estimator. Moreover, the time complexity of all the proposed estimators are the same as that of the state-of-the-art estimator. In addition, we also integrate an elegant graph simplification technique into the proposed estimators to further improve the accuracy and running time of our estimators. We conduct extensive experiments to evaluate the proposed estimators. The results demonstrate the effectiveness, efficiency, and scalability of our estimators.

Acknowledgements

The work was supported in part by (i) NSFC Grants (61402292, 61170076, U1301252, 61033009) and Natural Science Foundation of SZU (grant no. 201438); (ii) Research Grants Council of the Hong Kong SAR, China, 14209314 and 418512; (iii) China 863 (no. 2012AA010239) and Guangdong Key Laboratory Project (2012A061400024); (iv) National Key Technology Research and Development Program of the Ministry of Science and Technology of China (2014BAH28F05). Rui Mao is a corresponding author.

References

[1] Z. Zou, H. Gao, and J. Li, “Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics,” in *KDD*, 2010, pp. 633–642.

[2] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, “k-nearest neighbors in uncertain graphs,” *PVLDB*, vol. 3, no. 1, pp. 997–1008, 2010.

[3] R. Jin, L. Liu, B. Ding, and H. Wang, “Distance-constraint reachability computation in uncertain graphs,” *PVLDB*, vol. 4, no. 9, pp. 551–562, 2011.

[4] R. Jin, L. Liu, and C. C. Aggarwal, “Discovering highly reliable subgraphs in uncertain graphs,” in *KDD*, 2011.

[5] J. Bader, A. Chaudhuri, J. M. Rothberg, and J. Chant, “Gaining confidence in high-throughput protein interaction networks,” *Nature Biotechnology*, vol. 22, no. 1, pp. 78–85, 2003.

[6] S. Asthana, O. D. King, F. D. Gibbons, and F. P. Roth, “Predicting protein complex membership using probabilistic network reliability,” *Genome Research*, vol. 14, no. 6, pp. 1170–1175, 2004.

[7] 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*, 2007.

[8] D. Kempe, J. M. Kleinberg, and É. Tardos, “Maximizing the spread of influence through a social network,” in *KDD*, 2003.

[9] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, “Learning influence probabilities in social networks,” in *WSDM*, 2010.

[10] G. Rubino, “Network reliability evaluation,” pp. 275–302, 1999.

[11] D. Kempe, J. M. Kleinberg, and É. Tardos, “Influential nodes in a diffusion model for social networks,” in *ICALP*, 2005.

[12] S. K. Thompson, *Sampling*. Wiley-Interscience; 2 edition, 2002.

[13] R. Li, J. X. Yu, R. Mao, and T. Jin, “Efficient and accurate query evaluation on uncertain graphs via recursive stratified sampling,” in *ICDE*, 2014, pp. 892–903.

[14] Y. Zhou, H. Cheng, and J. X. Yu, “Clustering large attributed graphs: An efficient incremental approach,” in *ICDM*, 2010, pp. 689–698.

[15] T. Opsahl and P. Panzarasa, “Clustering in weighted networks,” *Social Networks*, vol. 31, no. 2, pp. 155–163, 2009.

[16] M. Newman, “The structure of scientific collaboration networks,” *Proc. Natl. Acad. Sci. USA*, vol. 98, pp. 404–409, 2001.

[17] P. Sevon, L. Eronen, P. Hintsanen, K. Kulovesi, and H. Toivonen, “Link discovery in graphs derived from biological databases,” in *DILS*, 2006.

[18] P. Hintsanen and H. Toivonen, “Finding reliable subgraphs from large probabilistic graphs,” *Data Min. Knowl. Discov.*, vol. 17, no. 1, pp. 3–23, 2008.

[19] Y. Yuan, G. Wang, H. Wang, and L. Chen, “Efficient subgraph search over large uncertain graphs,” *PVLDB*, vol. 4, no. 11, pp. 876–886, 2011.

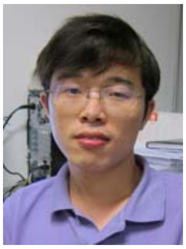
[20] A. Khan, F. Bonchi, A. Gionis, and F. Gullo, “Fast reliability search in uncertain graphs,” in *EDBT*, 2014.

[21] 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, 2010.

[22] L. Liu, R. Jin, C. C. Aggarwal, and Y. Shen, “Reliable clustering on uncertain graphs,” in *ICDM*, 2012.

[23] W. E. Moustafa, A. Kimmig, A. Deshpande, and L. Getoor, “Subgraph pattern matching over uncertain graphs with identity linkage uncertainty,” in *ICDE*, 2014.

[24] P. Parchas, F. Gullo, D. Papadias, and F. Bonchi, “The pursuit of a good possible world: extracting representative instances of uncertain graphs,” in *SIGMOD*, 2014, pp. 967–978.



Rong-Hua Li received the Ph.D. degree from the Chinese University of Hong Kong in 2013. He is currently an Assistant Professor at Shenzhen University, China. His research interests include algorithmic aspects of social network analysis, graph data management and mining, as well as sequence data management and mining.

Jeffery Xu Yu received the B.E., M.E., and Ph.D. degrees in computer science from the University of Tsukuba, Japan, in 1985, 1987, and 1990, respectively. He has held teaching positions at the Institute of Information Sciences and Electronics, University of Tsukuba, and at the Department of Computer Science, Australian National University, Australia. Currently, he is a Professor in the Department of Systems Engineering and Engineering Management, the Chinese University of Hong Kong, Hong Kong. He is serving as a VLDB Journal Editorial Board Member. His current research interests include graph database, graph mining, keyword search in relational databases, and social network analysis.

Rui Mao received the Ph.D. degree in computer science from the University of Texas at Austin, TX, USA, in 2007. He is currently an Associate Professor at Shenzhen University, China. His current research interests include big data analysis and management, content-based similarity query of multimedia and biological data, and data mining.

Tan Jin received the Ph.D. degree from the Chinese University of Hong Kong in 2012. He is currently an Associate Professor at Northeastern University, China. His current research interests include linguistic data mining, data management, and computational linguistic.