# Efficient and Accurate Query Evaluation on Uncertain Graphs via Recursive Stratified Sampling

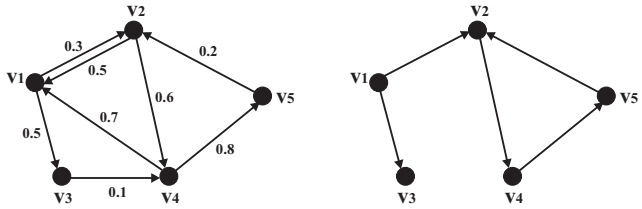
Rong-Hua Li [#], Jeffrey Xu Yu [*†], Rui Mao [#], and Tan Jin [◇]

[#] *Guangdong Province Key Laboratory of Popular High Performance Computers, Shenzhen University, China*
[*] *The Chinese University of Hong Kong,* [◇] *Northeastern University, China*
[†] *Key Laboratory of High confidence Software Technologies Ministry of Education (CUHK Sub-Lab)*
{rhli, yu}@se.cuhk.edu.hk, mao@szu.edu.cn, alextanjin@gmail.com

***Abstract*—In this paper, we introduce two types of query evaluation problems on uncertain graphs: expectation query evaluation and threshold query evaluation. Since these two problems are #P-complete, most previous solutions for these problems are based on naive Monte-Carlo (*NMC*) sampling. However, *NMC* typically leads to a large variance, which significantly reduces its effectiveness. To overcome this problem, we propose two classes of estimators, called class-I and class-II estimators, based on the idea of stratified sampling. More specifically, we first propose two classes of basic stratified sampling estimators, named *BSS-I* and *BSS-II*, which partition the entire population into $2^r$ and $r+1$ strata by picking $r$ edges respectively. Second, to reduce the variance, we find that both *BSS-I* and *BSS-II* can be recursively performed in each stratum. Therefore, we propose two classes of recursive stratified sampling estimators called *RSS-I* and *RSS-II* respectively. Third, for a particular kind of problem, we propose two *cut-set* based stratified sampling estimators, named *BCSS* and *RCSS*, to further improve the accuracy of the class-I and class-II estimators. For all the proposed estimators, we prove that they 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 the time complexity of *NMC* under a mild assumption. In addition, we also apply the proposed estimators to influence function evaluation and expected-reliable distance query problem, which are two instances of the query evaluation problems on uncertain graphs. Finally, we conduct extensive experiments to evaluate our estimators, and the results demonstrate the efficiency, accuracy, and scalability of the proposed estimators.**

## I. 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].




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

Fig. 1. Running example

In uncertain graph management, a fundamental problem is to evaluate the queries efficiently and accurately. In this paper, we formulate two types of query evaluation problems: the expectation query evaluation and the 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 $\delta$. Many applications on uncertain graph management can be formulated as the above 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 the 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

∗ Dr. Rui Mao is the corresponding author.

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.

In general, all the above mentioned expectation and 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 has 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 Monte-Carlo estimators, named class-I and class-II estimators, based on the idea of stratified sampling. Specifically, for the class-I estimators, we first propose a basic stratified sampling estimator called *BSS-I*. *BSS-I* partitions the probability space $\Omega$ (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*. Moreover, we find that *BSS-I* can be recursively applied in each stratum, and thereby we propose a recursive stratified sampling estimator, named *RSS-I* estimator. Since *RSS-I* recursively reduces the variance in each stratum, its variance is significantly smaller than that of *BSS-I*. For the class-II estimators, we also propose a basic stratified sampling estimator (*BSS-II*) and a recursive stratified sampling estimator (*RSS-II*). The idea of the class-II estimators is similar to the idea of the class-I estimators. The major difference is the stratification method used. In particular, for the class-II estimators, we develop a new stratification method. This new stratification method splits the probability space $\Omega$ into $r+1$ strata by selecting $r$ edges. Compared to the stratification method used in the class-I estimators, the advantage of this method is that we can finely tune the number of strata of the class-II estimators by tuning the parameter $r$ because it only produces $r+1$ strata. Both the class-I and class-II estimators are shown to be unbiased and their variances are significantly smaller than that of *NMC*. Additionally, an important property of both class-I and class-II estimators is that they have the same time complexity as *NMC* under a mild assumption, satisfying in most real-world applications. That is to say, the class-I and class-II estimators improve the accuracy of the *NMC* estimator without scarifying efficiency.

The proposed class-I and class-II estimators are very general which can be used as a building block for any query evaluation on uncertain graph problem. However, these methods do not capture 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 V. 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 apply the proposed estimators to the influence function evaluation problem and the expected-reliable distance query problem which are two instances of our query evaluation problems. Finally, we perform extensive experiments to evaluate the proposed estimators. The results show that our class-I and class-II estimators significantly outperform the state-of-the-art estimator. Moreover, we find that the *cut-set* based estimators can significantly improve the accuracy of the class-I and class-II estimators. The results also show that all of the proposed estimators scale linearly w.r.t. the graph size, thus all of them can be used to handle large graphs.

## II. 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

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

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

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

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

*Definition 2.2:* **(Threshold query evaluation)** Given a threshold $\delta$, the threshold query evaluation problem is a

[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.

problem of calculating the following expected value

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

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

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

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 $\delta$. 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 I, many applications of uncertain graph management and mining 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 the above mentioned 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

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

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.

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

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.

TABLE I
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 | $*$ | $\cdots$ | $*$ | $\Omega_1$ |
| Stratum 2 | 1 | 0 | 0 | $\cdots$ | 0 | $*$ | $\cdots$ | $*$ | $\Omega_2$ |
| Stratum 3 | 0 | 1 | 0 | $\cdots$ | 0 | $*$ | $\cdots$ | $*$ | $\Omega_3$ |
| $\cdots$ | | | $\cdots$ | | | | | | $\cdots$ |
| Stratum $2^r$ | 1 | 1 | 1 | $\cdots$ | 1 | $*$ | $\cdots$ | $*$ | $\Omega_{2^r}$ |

## III. NEW CLASS-I ESTIMATORS

To reduce the variance of *NMC*, in this section, we propose two new estimators for expectation query evaluation. To distinguish the class-II estimators which we will present in Section IV, we refer to the new estimators proposed in this section as the class-I estimators. More specifically, we will present a new basic stratified sampling estimator, named *BSS-I*, in Section III-A, and propose a new recursive stratified sampling estimator, named *RSS-I*, in Section III-B.

### A. 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 $\Omega$ (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 I.

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 I, 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 $\Omega_1$ do not include the edges in $T$. The probability of a possible graph in Stratum $i$ $(i = 1, \cdots, 2^r)$ is given by

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

In our stratification approach, a question that arises is how to select the $r$ edges for stratification. As shown in the experiments, 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 have direct contribution to calculate $\phi_q(G)$. 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* (breadth-first-search) 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.

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

where $\pi_i$ is defined in Eq. (7). The following theorem shows that $\hat{\Phi}_{BSSI}$ is an unbiased estimator of $\Phi_q(\mathcal{G})$. Due to space limit, all the proofs of this paper are omitted and they can be found in our technical report [13].

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

Let $\sigma_i$ 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

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

**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

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

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. $\sigma_i$, 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*.

*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 outline the *BSS-I* algorithm in Algorithm 1. Assume that computing the query evaluation function for each possible graph takes $O(M)$ time (line 12). Then, one can easily derive that the time complexity of Algorithm 1 is $O(N(m + M))$ which is equal to the time complexity of the *NMC* algorithm.

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

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

```
1:  Φ̂ ← 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;
5:      Compute π_i by Eq. (7);
6:      N_i ← [π_i N];
7:      t ← 0;
8:      for j = 1 to N_i do
9:          Flip m − r coins to determine the rest m − r edges;
10:         Let Y_j be the status vector of the rest m − r edges;
11:         Append X_i to Y_j to generate a possible graph G_j;
12:         Compute φ_q(G_j);
13:         t ← t + φ_q(G_j);
14:     t ← t/N_i;
15:     Φ̂ ← Φ̂ + π_i t;
16: return Φ̂;
```

### *B. 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 (*RSS-I*). The *RSS-I* algorithm is given in Algorithm 2. *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 $i$ (line 9-19 in Algorithm 2). Note that 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*.

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

The *RSS-I* algorithm terminates until the sample size is smaller than a given threshold ($\tau$) or the number of unsampled 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 $\Phi_q(\mathcal{G})$ (line 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.

We use the recursive tree technique [14] to analyze the time complexity of Algorithm 2. For sampling a possible graph, Algorithm 2 needs to traverse the recursive tree from the root node to the terminative node. Here the terminative node is a node in the recursive tree where the terminative conditions of the recursion satisfy at that node, i.e., $N < \tau$ or $|E_2| < r$ holds in Algorithm 2. Let $\bar{d}$ be the average length of the path from the root node to the terminative node. Then, at each internal node of the path, the time complexity is $O(r)$. Suppose that the total number of such paths is $K$. Then, the algorithm takes $O(K\bar{d}r)$ time at the internal nodes of all the paths. Note that $K$ is bounded by the sample size $N$, and $\bar{d}$ is a very small number w.r.t. $N$. More specifically, we can derive that $\bar{d} = O(\log_{2^r} N)$, which is a very small number. For example, assume that $r = 5$ and $N = 100,000$, then we can get $\bar{d} \approx 3.3$. For all the terminative nodes, the time complexity of the algorithm is $O(N(m+M))$. This is because the algorithm needs to sample $N$ possible graphs in total over all the terminative nodes, and for each possible graph the algorithm has to compute the query evaluation function which takes $O(M)$ time. Since $O(K\bar{d}r)$ is dominated by $O(Nm)$, the time complexity of Algorithm 2 is $O(N(m+M) + K\bar{d}r) = O(N(m+M))$.

**Algorithm 2** *RSS-I* ($\mathcal{G}$, $E_1$, $E_2$, $X$, $N$, $q$, $r$, $\tau$)

**Input**: An uncertain graph $\mathcal{G} = (V, E, P)$, the set of sampled edges $E_1$, the set of unsampled edges $E_2$, 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 $|E_2| < r$ **then**
3: **for** $j = 1$ to $N$ **do**
4: Flip $|E_2|$ coins 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: Select $r$ edges from $E_2$ by an edge-selection strategy;
10: Let $T$ be the set of selected edges;
11: **for** $i = 1$ to $2^r$ **do**
12: $Y \leftarrow X$ {Recording the current status vector $X$};
13: Let $X_i$ be the status vector of set $T$ in Stratum $i$;
14: Append $X_i$ to $Y$;
15: Compute $\pi_i$ by Eq. (7);
16: $N_i \leftarrow [\pi_i N]$;
17: $\mu_i \leftarrow$ *RSS-I* ($\mathcal{G}$, $E_1 \cup T$, $E_2 \backslash T$, $Y$, $N_i$, $q$, $r$, $\tau$);
18: $\hat{\Phi} \leftarrow \hat{\Phi} + \pi_i \mu_i$;
19: **return** $\hat{\Phi}$;

## IV. New Class-II Estimators

In this section, we propose two new stratified sampling estimators, named class-II basic stratified sampling estimator (*BSS-II*) and class-II recursive stratified sampling estimator (*RSS-II*). Below, we introduce these estimators in detail.

### A. Basic stratified sampling (BSS-II)

**Stratification**: In *BSS-II*, the new stratification method splits the probability space $\Omega$ 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 stratification method used in *BSS-I*, this new stratification approach allows us to set $r$ to be a relatively large number, such as $r = 100$. Furthermore, we can 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. The stratum design method of the *BSS-II* estimator is depicted in Table II.

TABLE II
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 | $*$ | $\cdots$ | $*$ | $\Omega_0$ |
| Stratum 1 | 1 | $*$ | $*$ | $\cdots$ | $*$ | $*$ | $\cdots$ | $*$ | $\Omega_1$ |
| Stratum 2 | 0 | 1 | $*$ | $\cdots$ | $*$ | $*$ | $\cdots$ | $*$ | $\Omega_2$ |
| Stratum 3 | 0 | 0 | 1 | $\cdots$ | $*$ | $*$ | $\cdots$ | $*$ | $\Omega_3$ |
| $\cdots$ | | | | $\cdots$ | | | | | $\cdots$ |
| Stratum $r$ | 0 | 0 | 0 | $\cdots$ | 1 | $*$ | $\cdots$ | $*$ | $\Omega_r$ |

In Table II, 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

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

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, we conclude that the stratum design approach described in Table II 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

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

where $\pi_i'$ 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

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

where $\sigma_i$ 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 can easily present the *BSS-II* algorithm. Due to space limit, we omit the details and refer the readers to [13]. It is worth mentioning that the edge-selection strategies used in the *BSS-I* algorithm can also be used in the *BSS-II* algorithm. In addition, one can easily derive that the time complexity of *BSS-II* is $O(N(m+M))$.

### B. 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 $\Omega$ into $r+1$ subspace $\Omega_i$ $(i = 0, 1, \cdots, r)$ by the stratification method used in *BSS-II*. Then, the same partition procedure is recursively performed in each subspace $\Omega_i$. In each partition, *RSS-II* utilizes the sample allocation method used in *BSS-II* to allocate the sample size. The recursion process of *RSS-II* will terminate until the sample size is smaller than a given threshold ($\tau$) or the number of unsampled 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 $O(N(m+M))$ using the recursive tree technique. The detailed description of the *RSS-II* algorithm and the time complexity analysis can be found in [13].

## V. 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.

### A. 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 $\Omega$ into two subspaces denoted by $\Omega_0$ and $\bar{\Omega}_0$ based on the cut-set $C$. Here $\Omega_0$ 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 $\bar{\Omega}_0 = \Omega \backslash \Omega_0$.

TABLE III
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 | $*$ | $*$ | $\cdots$ | $*$ | $*$ | $\cdots$ | $*$ | $\Omega_1$ |
| Stratum 2: | 0 | 1 | $*$ | $\cdots$ | $*$ | $*$ | $\cdots$ | $*$ | $\Omega_2$ |
| Stratum 3: | 0 | 0 | 1 | $\cdots$ | $*$ | $*$ | $\cdots$ | $*$ | $\Omega_3$ |
| $\cdots$ | | | | $\cdots$ | | | | | $\cdots$ |
| Stratum $\lvert C\rvert$ : | 0 | 0 | 0 | $\cdots$ | 1 | $*$ | $\cdots$ | $*$ | $\Omega_{\lvert C\rvert}$ |

Let $C = \{e_1, e_2, \cdots, e_{|C|}\}$, then the probability of a possible graph $G$ in $\Omega_0$ is given by

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

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 $\Omega_0$. Instead, we can focus on picking samples from $\bar{\Omega}_0$. Based on this idea, we propose the focal sampling (FS) estimator $\hat{\Phi}_{FS}$ as follows

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

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

Let $\bar{u}_0$ and $\bar{\sigma}_0$ be the expectation and variance of the samples in $\bar{\Omega}_0$ respectively. Theorem 5.3 shows that the variance of the FS estimator ($\hat{\Phi}_{FS}$) is no larger than that of the *NMC* estimator.

*Theorem 5.3:* $\mathrm{var}(\hat{\Phi}_{FS}) \leq \mathrm{var}(\hat{\Phi}_{NMC})$.

By Theorem 5.2 and Theorem 5.3, we conclude that the FS estimator is a more accurate unbiased estimator than *NMC*.

### B. 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 (*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., $\Omega_i$ denotes Stratum $i$ for $i = 1, \cdots, |C|$, and draw samples separately from each stratum. The detailed stratification method is given in Table III.

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

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

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

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

where $\pi_0^c$ is given in Eq. (15).

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

**Input**: An uncertain graph $\mathcal{G} = (V, E, P)$, sample size $N$, and a query $q$

**Output**: The *BCSS* estimator $\hat{\Phi}$

1: $\hat{\Phi} \leftarrow 0$;
2: Compute the cut-set $C$ based on $q$ and $\phi_q(G)$;
3: **for** $i = 1$ to $|C|$ **do**
4: Compute $\pi_i^c$ (Eq. (17)) and $\pi_i^{cd}$ (Eq. (21));
5: $N_i \leftarrow [\pi_i^{cd} N]$;
6: $t \leftarrow 0$;
7: Let $E_i$ be the set of edges to be determined in Stratum $i$;
8: **for** $j = 1$ to $N_i$ **do**
9: Flip $m - i$ coins to determine $E_i$, and thus generate a possible graph $G_j$;
10: Compute $\phi_q(G_j)$;
11: $t \leftarrow t + \phi_q(G_j)$;
12: $t \leftarrow t/N_i$;
13: $\hat{\Phi} \leftarrow \hat{\Phi} + \pi_i^c t$;
14: Calculate $\pi_0$ by Eq. (15) and $u_0$;
15: **return** $\hat{\Phi} + \pi_0^c u_0$;

**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

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

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

*Theorem 5.4:* $\mathbb{E}(\hat{\Phi}_{BCSS}) = \Phi_q(\mathcal{G})$.

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

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

where $\sigma_i$ 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

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

where $i = 1, \cdots, |C|$ and $\Pr[G \in \Omega_i | G \notin \Omega_0]$ denotes the probability of sampling a possible graph from $\Omega_i$ conditioning on it is not in $\Omega_0$. 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 5.5:* $\mathrm{var}(\hat{\Phi}_{BCSS}) \leq \mathrm{var}(\hat{\Phi}_{FS})$.

The *BCSS* algorithm is outlined in Algorithm 3. The time complexity of Algorithm 3 is $O(N(m+M)+T)$, where computing the cut-set takes $O(T)$ time. The detailed description of the algorithm and the time complexity analysis can be found in [13]. 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*.

**Algorithm 4** RCSS($\mathcal{G}$, $E_1$, $E_2$, $X$, $N$, $S$, $q$)

**Input**: An uncertain graph $\mathcal{G} = (V, E, P)$, the set of sampled edges $E_1$, the set of unsampled edges $E_2$, the status vector of the sampled edges $X$, the sample size $N$, the answer set $S$, and the query $q$

**Output**: The *RCSS* estimator $\hat{\Phi}$

1: $\hat{\Phi} \leftarrow 0$;
2: Compute the cut-set $C'$ based on $S$, $q$ and $\phi_q(G)$;
3: Let $C = (e_1, e_2, \cdots, e_{|C|}) = C' \bigcap E_2$;
4: **if** $N < \tau_1$ or $|E_2| < \tau_2$ or $|C| == 0$ **then**
5: **for** $j = 1$ to $N$ **do**
6: Flip $|E_2|$ coins to generate a possible graph $G_j$;
7: Compute $\phi_q(G_j)$;
8: $\hat{\Phi} \leftarrow \hat{\Phi} + \phi_q(G_j)$;
9: **return** $\hat{\Phi}/N$;
10: **else**
11: Compute $\pi_0^c$ and $u_0$ based on $C$ and $X$;
12: **for** $i = 1$ to $|C|$ **do**
13: Compute $\pi_i^c$ (Eq. (17)) and $\pi_i^{cd}$ (Eq. (21));
14: $N_i \leftarrow [\pi_i^{cd} N]$;
15: Let $T_i = \{e_1, \cdots, e_i\}$;
16: Let $X_i$ be the status vector of set $T_i$ under Stratum $i$;
17: Append $X_i$ to $X$;
18: $R \leftarrow S$; {Record the current answer set $S$}
19: Update $R$ if any;
20: $\mu_i \leftarrow$RCSS($\mathcal{G}$, $E_1 \cup T_i$, $E_2 \backslash T_i$, $X$, $N_i$, $R$, $q$);
21: $\hat{\Phi} \leftarrow \hat{\Phi}$ + $\pi_i^c \mu_i$;
22: **return** $\hat{\Phi}$ + $\pi_0^c \mu_0$;

### C. 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. Note that in each recursion of Algorithm 4, we maintain an answer set $S$ to record the immediate results obtained from the previous recursion (line 18-19), and it will be used to compute the cut-set (line 2). We will describe the function of the answer set $S$ by two concrete applications in Section V-E. 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 $O(N(M+m+T))$ by using the recursive tree. 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*. Indeed, in our experiments, we observe that the average query time of *RCSS* and *NMC* are comparable. Due to space limit, the detailed description of Algorithm 4 and its complexity analysis are given in [13].

### D. Discussion

Here we give a brief discussion of all the proposed estimators. First, recall that the stratification method of *BCSS* 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 5.6:* If $r = |C|$, $\mathrm{var}(\hat{\Phi}_{BCSS}) \leq \mathrm{var}(\hat{\Phi}_{BSSII})$.

Theorem 5.6 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 variance of *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 variance of *BCSS* and *BSS-I* and compare the variance of *RCSS*, *RSS-I*, and *RSS-II*. Notice that the class-I and class-II estimators are very general which do not depend on the graph structure and the property of 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 show that the performance of such data-driven methods are significantly better than those of the class-I and class-II estimators.

### *E. Two applications*

In this subsection, we introduce two applications, the influence function evaluation problem [8], [11] and the expected-reliable distance query problem [2] to illustrate the idea of *BCSS* and *RCSS* estimators. Note that the proposed class-I (*BSS-I* and *RSS-I*) and class-II (*BSS-II* and *RSS-II*) estimators can be directly used for these two query evaluation problems. Here we focus on using the *BCSS* and *RCSS* estimators for these problems.

**Influence function evaluation:** Given an uncertain graph $\mathcal{G}$ and a seed set $A$, the influence function evaluation problem is a problem of computing 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 very 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 influence function evaluation problem with seed set $A$ is equivalent to the problem with seed node $q$. Clearly, such a problem is an instance of our 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 influence function evaluation problem, here we also introduce the threshold influence function evaluation problem. In particular, given an uncertain graph $\mathcal{G}$, a seed node $q$, a threshold $\delta$, the threshold influence function evaluation problem aims at estimating the probability of $\phi_q(G) \geq \delta$ for a possible graph $G$. In the following, we focus on the influence function evaluation problem. The algorithm can be easily generalized to the threshold influence function evaluation problem because we only need to replace $\phi_q(G)$ by $\mathrm{I}(\phi_q(G) \geq \delta)$.

Recall that in the *BCSS* and *RCSS* estimators, the key issue is to find the cut-set. For the influence function evaluation problem, we define the cut-set as the set of all the outgoing edges of the query node $q$. This is because if all the outgoing edges of $q$ are failed then there is no node that is reachable from $q$, thereby $\phi_q(G)$ is a constant 0, satisfying the definition of cut-set. For example, in Fig. 1(a), assume that $q = v_1$, then the cut-set $C$ is $\{v_1 \rightarrow v_2, v_1 \rightarrow v_3\}$. Note that the cut-set is not unique. For the influence function evaluation problem, the set of all the outgoing edges is the minimal cut-set. Adding any one edge in such a set is still a cut-set if the resulting set is not the entire edge set $E$.

After defining the cut-set, we can apply the *BCSS* estimator to the influence function evaluation problem. We only need to modify two lines (line 2 and line 10) in Algorithm 3 for the influence function evaluation problem. Specifically, in line 2, we get the cut-set by extracting all the outgoing edges of $q$. In line 10, we compute $\phi_q(G_j)$ by the *BFS* algorithm. Notice that in this problem $u_0$ is equivalent to 0. The detailed algorithm is omitted for brevity. The time complexity of the *BCSS* estimator for the influence function evaluation problem is $O(Nm)$. The reason is because the time complexity for computing $\phi_q(G_j)$ and $C$ is dominated by $O(m)$. Below, we focus on using *RCSS* estimator for influence function evaluation.

According to Algorithm 4, the key problem in the *RCSS* estimator is how to calculate the cut-set $C$ and $u_0$ in each recursion. To overcome this issue, we resort to the answer set $S$ (an input parameter in Algorithm 4). For influence function evaluation, the answer set $S$ records both the query node $q$ and the nodes that are reachable from $q$ until the current recursion. Based on the answer set $S$, the cut-set $|C|$ can be easily determined by the union of all the unsampled outgoing edges of the nodes in $S$, i.e., $C = (\bigcup_{v \in S} O_v) \bigcap E_2$, where $O_v$ denotes the set of all the outgoing edges of node $v$ and $E_2$ denotes the unsampled edge set (the statuses of the edges in $E_2$ are "$*$"). We can easily derive that $u_0$ is equivalent to $|S| - 1$, because until current recursion there are $|S| - 1$ nodes that are reachable from $q$. Consider an example in Fig. 1(a), assume that the query node is $v_1$, the current recursion depth is 2, and the status vector of the sampled edges (i.e., $(v_1 \rightarrow v_2, v_1 \rightarrow v_3)$) is $X = (0, 1)$. Then, under the current recursion, the answer set $S$ equals to $\{v_1, v_3\}$, the unsampled edge set $E_2 = \{v_2 \rightarrow v_1, v_3 \rightarrow v_4, v_4 \rightarrow v_1, v_2 \rightarrow v_4, v_4 \rightarrow v_5, v_5 \rightarrow v_2\}$. Given this, the cut-set $C$ is $C = (\bigcup_{v \in S} O_v) \bigcap E_2 = \{v_3 \rightarrow v_4\}$, and $u_0 = |S| - 1 = 1$. Besides, in each recursion, we need to update the answer set $S$ for the next recursion. Recall that by our stratification method, there is only one edge in the cut-set that will exist in the next recursion. For convenience, we refer to such an edge as the active edge. Therefore, we only need to add the head endpoint node[2] of the active edge into the answer set $S$.

For the algorithm, we only need to modify three lines in Algorithm 4. Specifically, in line 2, we compute the cut-set $C'$

[2]Here we assume the graph is directed without loss of generality.

by extracting the union of all the unsampled outgoing edges of the nodes in $S$. In line 7, we compute $\phi_q(G_j)$ by the *BFS* algorithm. In line 19, we update the answer set by adding the head endpoint node of the active edge into the answer set. For computing $u_0$, we set it to $|S| - 1$. Due to space limit, the detailed algorithm is given in [13]. The time complexity of this algorithm is $O(Nm)$, because computing $\phi_q(G_j)$ and $C$ takes $O(m)$ time.

**Expected-reliable distance query:** The expected-reliable distance 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

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

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 $\Omega_\infty$ denotes the probability space in which $s$ cannot reach $t$, i.e., $\phi_{s,t}(G) = \infty$ for $G \in \Omega_\infty$. Besides the expected-reliable distance 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 $\delta$, the threshold expected-reliable distance query is to estimate the probability of $\phi_{s,t}(G) \leq \delta$ for a possible graph $G$. Since the algorithm for the threshold expected-reliable distance query is very similar to the algorithm for the expected-reliable distance query, we focus on the expected-reliable distance query.

For completeness, we describe the *NMC* estimator for $\Phi_{s,t}(\mathcal{G})$. In particular, the *NMC* estimator first draws $N$ possible graphs $G_1, \cdots, G_N$. Second, for each possible graph $G_i$, the *NMC* estimator computes the distance from $s$ to $t$ ($\phi_{s,t}(G_i)$) by the *BFS* algorithm[3]. Suppose without loss of generality that in the first $N'$ ($N' \leq N$) possible graphs the distance from $s$ to $t$ is finite and in the rest $N - N'$ possible graphs the distance is infinity. Then, the *NMC* estimator is given by $\hat{\Phi}_{NMC} = \sum_{i=1}^{N'} \phi_{s,t}(G_i)/N'$.

Below, we make use of the *BCSS* and *RCSS* estimators for estimating $\Phi_{s,t}(\mathcal{G})$. The key issue for the *BCSS* and *RCSS* estimator is to define the cut-set. For the *BCSS* estimator, we define the cut-set $C$ as the set of all the outgoing edges of node $s$. The reason is because if all the outgoing edges of $s$ are failed, then the distance from $s$ to $t$ is a constant which is infinity. Note that if the cut-set $C$ is an empty set, then we definitely know that $s$ cannot reach $t$, thereby we do not need to do any sampling process. Reconsider the example in Fig. 1(a), assume that $s = v_1$ and $t = v_5$, then the cut-set $C = \{v_1 \rightarrow v_2, v_1 \rightarrow v_3\}$. Clearly, if both $v_1 \rightarrow v_2$ and $v_1 \rightarrow v_3$ are failed, then node $v_1$ cannot reach node $v_5$. Based on the cut-set $C$, we can use the *BCSS* estimator for the expected-reliable distance query problem. The algorithm is very similar to Algorithm 3, a notable difference is that in line 15 of Algorithm 3 we do not need to add $\pi_0^c u_0$ to $\hat{\Phi}$, because $u_0 = \infty$. For brevity, we omit the detailed description of the algorithm. The time complexity of the *BCSS* estimator is $O(Nm)$, because computing the cut-set takes $O(|C|)$ time complexity and calculating $\phi_{s,t}(G)$ by the *BFS* algorithm takes $O(m)$ time complexity.

For the *RCSS* estimator, the important steps are to compute the cut-set $C$ and $u_0$ in each recursion. In the expected-reliable distance query problem, $u_0$ always equals $\infty$. For the cut-set $C$, we use the answer set $S$ to calculate it. Recall that in each recursion, according to our stratification method, there is only one new edge will exist in the next recursion. Similarly, we refer to such an edge as the active edge. We set the answer set $S$ be a set containing only one node which is the head endpoint node of the active edge. Based on the answer set $S$, we let the cut-set $C$ be the set of unsampled outgoing edges of the node in $S$, i.e., $C = O_S \bigcap E_2$, where $O_S$ denotes the set of all the outgoing edges of the node in $S$. Consider an example in Fig. 1(a), suppose that $s = v_1$, $t = v_5$, the current recursion depth is 2, and the status vector of the sampled edges (i.e., $(v_1 \rightarrow v_2, v_1 \rightarrow v_3)$) is $X = (0, 1)$. Then, in the current recursion, the answer set $S$ is $\{v_3\}$, the unsampled edge set $E_2 = \{v_2 \rightarrow v_1, v_3 \rightarrow v_4, v_4 \rightarrow v_1, v_2 \rightarrow v_4, v_4 \rightarrow v_5, v_5 \rightarrow v_2\}$. Given this, the cut-set $C$ is $C = O_S \bigcap E_2 = \{v_3 \rightarrow v_4\}$. In each recursion, we can easily update the answer set $S$ by the head endpoint node of the active edge. In this example, the active edge is $v_3 \rightarrow v_4$, and for the next recursion we update the answer set $S$ by $\{v_4\}$.

For the algorithm description, it is similar to that of Algorithm 4. Two major differences include: (1) computing the cut-set (in line 2 of Algorithm 4), and (2) updating the answer set (line 19 of Algorithm 4). Specifically, we compute the cut-set by extracting the unsampled outgoing edges of the node in $S$, and update the answer set by the head endpoint node of the active edge. Due to space limit, the detailed algorithm is given in [13]. Also, we can easily derive that the time complexity of this algorithm is $O(Nm)$. This is because the time complexity for computing $\phi_{s,t}(G)$ and calculating the cut-set $C$ is dominated by $O(m)$.

## VI. Experiments

### A. Experimental setup

**Datasets:** We use one synthetic dataset and three 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 three 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

[3]Here we assume that the graph is un-weighted and directed, thus we can use *BFS* to compute the shortest path.

TABLE IV
SUMMARY OF THE DATASETS

| Name | Nodes | Edges | Ref. |
|---|---|---|---|
| ER | 5,000 | 50,616 | [3] |
| Facebook | 1,899 | 20,296 | [16] |
| Condmat | 16,264 | 95,188 | [17] |
| DBLP | 78,648 | 376,515 | [15] |

(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 [15]. Table IV summarizes the detailed information of our datasets. To obtain the uncertain networks, for each real-world dataset, we generate the probabilities according to the same method used 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 the edge.

**Different estimators**: In our experiments, we compare 12 different estimators. The first two estimators are served as the baselines, and the last ten estimators are our proposed 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. In this paper, we generalize this estimator to arbitrary parameter $r$ (the *RSS-I* estimator), and apply the generalized estimator for the general query evaluation problem. Recall that beyond *RM* edge-selection strategy, we also propose a more accurate *RSS-I* estimator with *BFS* edge-selection strategy for a kind of query evaluation problem. (3) BSSIR and BSSIB, which are the *BSS-I* estimator with *RM* and *BFS* edge-selection respectively. (4) RSSIR and RSSIB, which are the *RSS-I* estimator with *RM* and *BFS* edge-selection respectively. (5) BSSIIR and RSSIIR, which are the *BSS-II* estimator with *RM* and *BFS* edge-selection respectively. (6) RSSIIR and RSSIIB, which are the *RSS-II* estimator with *RM* and *BFS* edge-selection respectively. (7) BCSS and RCSS, which are the cut-set based estimators (i.e., *BCSS* and *RCSS*).

**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 $\sigma_{NMC}$ 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

$$\sum_{i=1}^{500} (\hat{\Phi}_i(\mathcal{G}) - \bar{\Phi}_i(\mathcal{G}))^2/499,$$

in which $\bar{\Phi}_i(\mathcal{G})$ denotes the mean of $\hat{\Phi}_i(\mathcal{G})$ $(i = 1, \cdots, 500)$.

**Parameter settings and experimental environment:** Without specifically stated, in all the experiments, we set the parameters as follows. For all estimators, we set the sample size $N = 1000$. For the *BSS-I* and *RSS-I* estimators, we set $r = 5$, and for the *BSS-II* and *RSS-II* estimators, we set $r = 50$, because under this setting these estimators perform very well. In [13], we have studied the effect of $r$ in these estimators. For the threshold parameter $\tau$ in *RSS-I* and *RSS-II* estimators, we set $\tau = 10$, and for the threshold parameters in *RCSS* estimator we set $\tau_1 = 10$ and $\tau_2 = 10$. All the experiments are conducted on a Scientific Linux 6.0 workstation with 2xQuad-Core Intel(R) 2.66 GHz CPU, and 4G memory. All algorithms are implemented in C++.

### *B. Results on influence function evaluation*

We report the experimental results for influence function evaluation problem. For the results of threshold influence function evaluation, we refer readers to [13]. In all the experiments, we randomly generate 1000 query nodes, and the results are the average result over all the queries.

Table V depicts the accuracy of different estimators. As can be seen, the *RCSS* estimator (RCSS) significantly outperforms the other competitors over all the datasets. In general, for the proposed estimators, the recursive estimators (RSSIR, RSSIB, RSSIIR, RSSIIB, and RCSS) outperform the basic estimators (BSSIR, BSSIB, BSSIIR, BSSIIB, and BCSS). For the proposed class-I and class-II estimators, we can find that the estimators with *BFS* edge-selection strategy are better than the estimators with *RM* edge-selection strategy. These results are consistent with our analysis in Section III and Section IV. In addition, we observe that all of our recursive estimators are significantly better than the state-of-the-art RSSIR1 estimator. For example, in Condmat dataset, RSSIR, RSSIB, RSSIIR, RSSIIB, and RCSS cut the relative variance of RSSIR1 by 62.9%, 303.5%, 62.6%, 275.0% and 482.3% respectively. Moreover, we can find that the basic class-I and class-II estimators are slightly worse than the RSSIR1 estimator, but they are still significantly better than the NMC estimator. Interestingly, we find that the basic cut-set based stratified sampling estimator (BCSS) consistently outperforms the state-of-the-art RSSIR1 estimator, and it is even better than the recursive estimators with *RM* edge-selection strategy in most datasets. The reason could be that BCSS captures the graph structure information and the cut-set property of the query evaluation function while the recursive estimators with *RM* strategy do not incorporate any structure information of the graph. For the efficiency of different estimators, we report the results in Table VI. We can see that the average query time of all the estimators are comparable in each dataset. These results confirm the time complexity analysis presented in the previous sections.

### *C. Results on expected-reliable distance query*

Here we report the results of expected-reliable distance query. For the results of threshold expected-reliable distance query, we refer readers to [13]. In all the experiments, we randomly generate 1000 query node-pairs $(s, t)$, the results are the average result over all the queries.

TABLE V
INFLUENCE FUNCTION EVALUATION: COMPARISON OF RELATIVE VARIANCE (RV) OF DIFFERENT ESTIMATORS

| RV | NMC | RSSIR1 | BSSIR | BSSIB | RSSIR | RSSIB | BSSIIR | BSSIIB | RSSIIR | RSSIIB | BCSS | RCSS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ER | 1.000 | 0.672 | 0.943 | 0.894 | 0.340 | 0.205 | 0.932 | 0.904 | 0.351 | 0.206 | 0.218 | **0.153** |
| Facebook | 1.000 | 0.559 | 0.890 | 0.682 | 0.302 | 0.257 | 0.695 | 0.667 | 0.279 | 0.240 | 0.324 | **0.168** |
| Condmat | 1.000 | 0.795 | 0.907 | 0.853 | 0.488 | 0.197 | 0.855 | 0.842 | 0.489 | 0.212 | 0.242 | **0.137** |
| DBLP | 1.000 | 0.538 | 0.917 | 0.837 | 0.210 | 0.192 | 0.945 | 0.800 | 0.200 | 0.182 | 0.205 | **0.125** |

TABLE VI
INFLUENCE FUNCTION EVALUATION: COMPARISON OF AVERAGE QUERY TIME OF DIFFERENT ESTIMATORS (IN SECOND)

| Time | NMC | RSSIR1 | BSSIR | BSSIB | RSSIR | RSSIB | BSSIIR | BSSIIB | RSSIIR | RSSIIB | BCSS | RCSS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ER | 0.359 | 0.356 | 0.350 | 0.375 | 0.337 | 0.378 | 0.363 | 0.375 | 0.372 | 0.385 | 0.368 | 0.396 |
| Facebook | 0.201 | 0.201 | 0.233 | 0.235 | 0.200 | 0.201 | 0.225 | 0.228 | 0.203 | 0.204 | 0.216 | 0.235 |
| Condmat | 1.297 | 1.296 | 1.304 | 1.305 | 1.205 | 1.241 | 1.251 | 1.310 | 1.226 | 1.228 | 1.297 | 1.306 |
| DBLP | 8.582 | 8.654 | 8.629 | 8.817 | 8.383 | 8.593 | 8.883 | 9.131 | 8.684 | 8.705 | 9.583 | 9.628 |

The expected-reliable distance query results are described in Table VII. From Table VII, we can see that RCSS achieves the best performance, followed by the proposed recursive estimators (RSSIR, RSSIB, RSSIIR, and RSSIIB), RSSIR1, the proposed basic estimators (BCSS, BSSIR, BSSIB, BSSIIR, and BSSIIB), and NMC. Similar to the results of the influence function evaluation, the proposed recursive estimators with *BFS* edge-selection strategy (RSSIB and RSSIIB) outperform the recursive estimators with *RM* strategy. It is also worth mentioning that all of our recursive estimators are significantly better than the state-of-the-art RSSIR1 estimator. For example, in Condmat dataset, RSSIR, RSSIB, RSSIIR, RSSIIB and RCSS reduce the relative variance of RSSIR1 by 16.3%, 17.3%, 15.6%, 18.0%, and 132.9% respectively. In addition, from Table VIII, we can observe that the average query time of all the estimators are comparable. These results further confirm our theoretical analysis in the previous sections.

### D. Scalability

To study the scalability of various estimators, 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) according to the same parameter setting described in Section VI-A. Also, for each estimator, we set the same parameter setting as our previous experiments. Fig. 2(a) and Fig. 2(b) depict the average query time of various estimators for the influence function evaluation and expected-reliable distance query respectively. In Fig. 2, the two numbers in the horizontal axis (eg. 200k/800k) denote the number of nodes and the number of edges respectively. From Fig. 2, we find that the average query time increases as the graph size increases. Generally, the average query time of all the estimators are comparable, and all the estimators exhibit a linear growth w.r.t. the graph size. These results demonstrate that all the estimators scale linearly w.r.t. the graph size, which are consistent with the time complexity of the estimators.

### E. Effect of sample size

As shown in the previous experiments, RCSS, RSSIB, and RSSIIB outperform the other estimators. Here we study how the sample size affects the estimating accuracy of these estimators in Condmat dataset. Similar results can also be

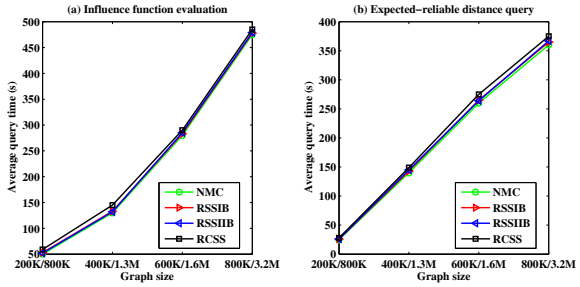



Fig. 2. Scalability testing

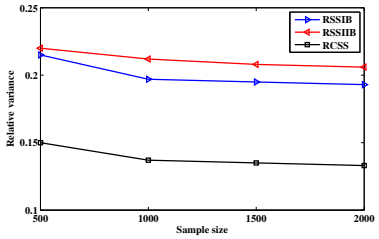



Fig. 3. Relative variance vs. sample size

observed in the other datasets. Fig. 3 shows the relative variance of the estimators under different sample sizes. As can be observed, the curves of RCSS, RSSIB, and RSSIIB are very smooth when the sample size is no less than 1000, which indicate that the relative variance of these estimators are relatively robust w.r.t. the sample size.

## VII. RELATED WORK

Uncertain graph management and mining has been attracted much attention due to the increasing applications in biological database [18], communication networks [7], and influence networks [8]. Notable work on uncertain graph management and mining consists of finding the reliable subgraph in a large uncertain graph [19], [4], frequent subgraph mining in uncertain graph database [1], [20], subgraph search in large uncertain graph [21], K-nearest neighbor search in uncertain graph [2], and distance constraint reachability computation in uncertain graph [3]. Generally, all the mentioned uncertain graph problems are known to be #P-complete, thus finding the exact solution is impossible in large uncertain graphs. Therefore, most existing work, 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. In this paper, we develop several accurate estimators without

TABLE VII
EXPECTED-RELIABLE DISTANCE QUERY: COMPARISON OF RELATIVE VARIANCE (RV) OF DIFFERENT ESTIMATORS

| RV | NMC | RSSIR1 | BSSIR | BSSIB | RSSIR | RSSIB | BSSIIR | BSSIIB | RSSIIR | RSSIIB | BCSS | RCSS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ER | 1.000 | 0.772 | 0.921 | 0.913 | 0.682 | 0.679 | 0.923 | 0.920 | 0.678 | 0.675 | 0.908 | **0.398** |
| Facebook | 1.000 | 0.759 | 0.932 | 0.925 | 0.651 | 0.648 | 0.934 | 0.931 | 0.653 | 0.650 | 0.921 | **0.356** |
| Condmat | 1.000 | 0.815 | 0.916 | 0.909 | 0.701 | 0.695 | 0.912 | 0.909 | 0.705 | 0.691 | 0.903 | **0.350** |
| DBLP | 1.000 | 0.756 | 0.924 | 0.917 | 0.697 | 0.683 | 0.920 | 0.915 | 0.689 | 0.680 | 0.901 | **0.515** |

TABLE VIII
EXPECTED-RELIABLE DISTANCE QUERY: COMPARISON OF AVERAGE QUERY TIME OF DIFFERENT ESTIMATORS (IN SECOND)

| Time | NMC | RSSIR1 | BSSIR | BSSIB | RSSIR | RSSIB | BSSIIR | BSSIIB | RSSIIR | RSSIIB | BCSS | RCSS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ER | 0.405 | 0.422 | 0.408 | 0.410 | 0.428 | 0.431 | 0.406 | 0.410 | 0.425 | 0.427 | 0.412 | 0.453 |
| Facebook | 0.210 | 0.231 | 0.214 | 0.216 | 0.235 | 0.216 | 0.217 | 0.233 | 0.235 | 0.236 | 0.218 | 0.241 |
| Condmat | 1.383 | 1.387 | 1.385 | 1.383 | 1.389 | 1.401 | 1.386 | 1.388 | 1.405 | 1.408 | 1.400 | 1.410 |
| DBLP | 11.33 | 11.41 | 11.35 | 11.37 | 11.45 | 11.46 | 11.35 | 11.36 | 11.40 | 11.43 | 11.40 | 11.48 |

sacrificing efficiency for the query evaluation problems on uncertain graphs.

Our work is also related to the query evaluation problem in uncertain database [22]. In [23], Dalvi, et al. proposed a framework to support arbitrarily SQL query with uncertain predicates. Subsequently, in [24], Christopher, et al. proposed a Monte-Carlo based method for top-K query evaluation on uncertain database. More recently, Li, et al. [25] proposed a unified approach for top-K query processing in uncertain database. Note that the techniques for query evaluation in uncertain database cannot be directly used in uncertain graphs. The reason is as follows. Generally, in uncertain database, sampling a possible word takes constant time complexity, while in uncertain graphs, sampling a possible graph is a time-consuming step which takes $O(m)$ time complexity. Therefore, for the query evaluation in uncertain graphs, the key issue is to reduce the number of samples for the estimators. To reduce the sample size of the estimator, one effective solution is to reduce its variance. To that end, in this paper, we propose several recursive stratified sampling estimators, and we show that the variance of our estimators are significantly smaller than those of the existing estimators.

## VIII. CONCLUSION

In this paper, we introduce two types of query evaluation problems on uncertain graphs. To solve these problems, we propose several efficient and accurate estimators based on stratified sampling. We show that all of our 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. 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) grant of the Research Grants Council of Hong Kong SAR, China No. CUHK/ 418512, (ii) NSFC project 61170076, and (iii) China-863 project 2012AA01A309.

## 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.-H. Li, J. X. Yu, R. Mao, and T. Jin, "Efficient and accurate query evaluation on uncertain graphs via recursive stratified sampling," *Technical Report*, Available at http://nhpcc.szu.edu.cn/icdeugsampler.pdf, 2013.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press; third edition, 2009.

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

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

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

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

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

[20] 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.

[21] 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.

[22] D. Suciu, D. Olteanu, C. Re, and C. Koch, *Probabilistic Databases*. Morgan & Claypool, 2011.

[23] N. N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," *VLDB J.*, vol. 16, no. 4, pp. 523–544, 2007.

[24] C. Re, N. N. Dalvi, and D. Suciu, "Efficient top-k query evaluation on probabilistic data," in *ICDE*, 2007.

[25] J. Li, B. Saha, and A. Deshpande, "A unified approach to ranking in probabilistic databases," *VLDB J.*, vol. 20, no. 2, pp. 249–275, 2011.