
EFFICIENT TOP-K VULNERABLE NODES DETECTION IN UNCERTAIN GRAPHS

Dawei Cheng

Chen Chen

Xiaoyang Wang

Sheng Xiang

ABSTRACT

Uncertain graphs have been widely used to model complex linked data in many real-world applications, such as guaranteed-loan networks and power grids, where a node or edge may be associated with a probability. In these networks, a node usually has a certain chance of default or breakdown due to self-factors or the influence from upstream nodes. For regulatory authorities and companies, it is critical to efficiently identify the vulnerable nodes, i.e., nodes with high default risks, such that they could pay more attention to these nodes for the purpose of risk management. In this paper, we propose and investigate the problem of top- k vulnerable nodes detection in uncertain graphs. We formally define the problem and prove its hardness. To identify the k most vulnerable nodes, a sampling-based approach is proposed. Rigorous theoretical analysis is conducted to bound the quality of returned results. Novel optimization techniques and a bottom- k sketch based approach are further developed in order to scale for large networks. In the experiments, we demonstrate the performance of proposed techniques on 3 real financial networks and 5 benchmark networks. The evaluation results show that the proposed methods can achieve up to 2 orders of magnitudes speedup compared with the baseline approach. Moreover, to further verify the advantages of our model in real-life scenarios, we integrate the proposed techniques with our current loan risk control system, which is deployed in the collaborated bank, for more evaluation. Particularly, we show that our proposed new model has superior performance on real-life guaranteed-loan network data, which can better predict the default risks of enterprises compared to the state-of-the-art techniques.

Keywords Uncertain graph · guaranteed-loan network · top- k · vulnerable node detection

1 Introduction

Uncertainty is inherent in real-world data because of various reasons, such as the accuracy issue of devices and models [1, 2]. Sometimes, people may inject uncertainty to the data on purpose to protect user privacy [3]. As a common data structure, graphs are widely used to model the complex relationships between different entities. Due to the ubiquitous uncertainty, uncertain graph analysis has attracted significant attentions in the community of database management. A large number of graph problems have been studied in the context of uncertain graphs, e.g., nearest neighbor search [4], reliability query [5], cohesive subgraph mining [1], etc.

In some real-life graphs, such as power grids and guaranteed-loan networks, nodes (e.g., facilities and enterprises) may breakdown or default due to self-factors or the issues from upstream nodes. To identify these high-risky (i.e., vulnerable) nodes, in this paper, we investigate a novel problem, named top- k vulnerable nodes detection. Given a directed uncertain graph \mathcal{G} , each node A (resp. edge (A, B)) is associated with a probability $p_s(A)$ (resp. $p(B|A)$). $p_s(A)$ denotes the probability that A defaults due to itself factors, and $p(B|A)$ denotes the probability that B defaults because of the default of A . By considering both factors, we can calculate the node default probability. We say a node is more vulnerable if it has higher default probability. The problem is different from the existing research, such as reliability problem and influence maximization problem [5, 6], which more focus on investigating the reachability for a set of nodes or finding a group of nodes to maximize the influence over the network. Our problem is of great importance to many real-world applications. Following is a motivating example on financial data analysis.

Motivating application. Network-guaranteed loan (also known as guarantee circle) is a widespread economic phenomenon, and attracting increasing attention from the banks, financial regulatory authorities, governments, etc. In order

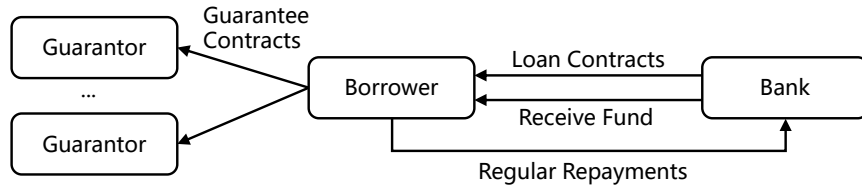


Figure 1: Guarantee loan process

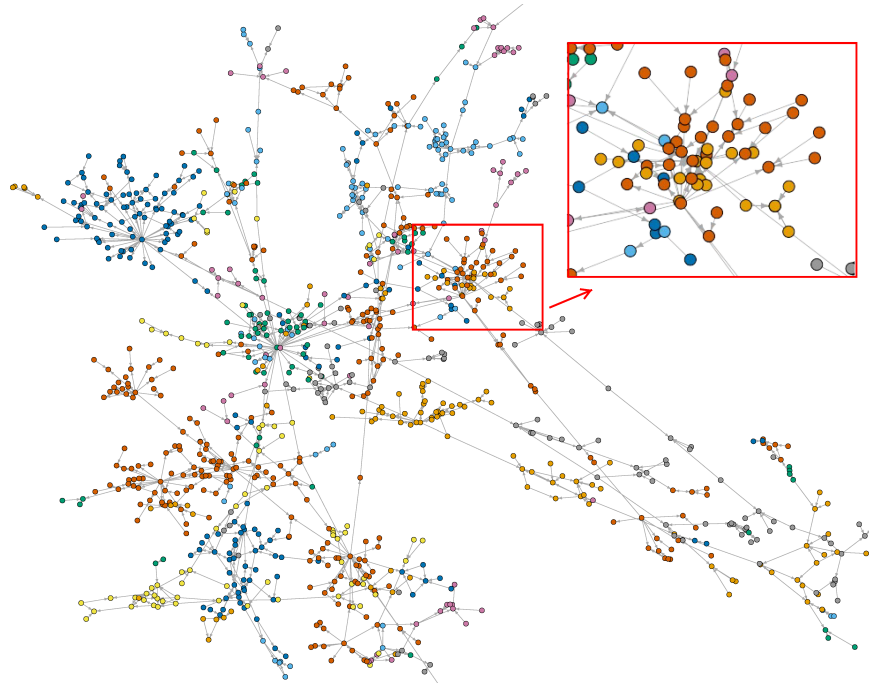


Figure 2: A real-world guaranteed-loan network

to obtain loans from banks, groups of small and medium enterprises (SMEs) back each other to enhance their financial security. Figure 1 shows the flow of guarantee loan procedure. When more and more enterprises are involved, they form complex directed-network structures [7]. Figure 2 illustrates a guaranteed-loan network with around 3,000 enterprises and 7,000 guarantee relations, where a node represents a small or medium enterprise and a directed edge from node A to node B indicates that enterprise B guarantees another enterprise A . Thousands of guaranteed-loan networks of different complexities have coexisted for a long period [8]. It requires an efficient strategy to prevent, identify and dismantle systematic crises.

Many kinds of risk factors have emerged throughout the guaranteed-loan network, which may accelerate the transmission and amplification of risk. The guarantee network may be alienated from the “mutual aid group” as a “breach of contract”. An appropriate guarantee union may reduce the default risk, but significant contagion damage throughout the networked enterprises could still occur in practice [9]. The guaranteed loan is a debt obligation promise. If one corporation gets trapped in risks, it will spread the contagion to other corporations in the network. When defaults diffuse across the network, a systemic financial crisis may occur. It is essential to consider the contagion damage in the guaranteed-loan networks. We can model a guaranteed-loan network with our uncertain graph model, where each node has a self-risk probability and each edge has a risk diffusion probability. Figure 3(a) is toy example of guaranteed-loan network, where Figure 3(e) shows the corresponding relationships between different enterprises and the risk diffusion probabilities. It is desirable to efficiently identify the k most vulnerable nodes, i.e., enterprises with high default risks, such that the banks or the financial regulatory authorities can pay extra attention to them for the purpose of risk management. It is more urgent than ever before with the slowdown of the economics worldwide nowadays.

In the literature, some machine-learning based approaches (e.g., [10]) have been proposed to predict the node default risk for different applications. For instance, a high-order and graph attention based network representation method has been designed in [10] to infer the possibility of loan default events. These approaches indeed consider the structure of

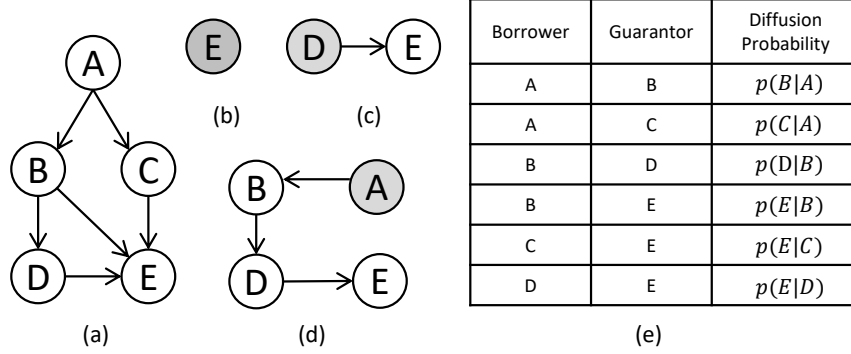


Figure 3: Example of uncertain guaranteed-loan network

networks. However, they cannot properly capture the uncertain nature of contagious behaviors in the networks. In the experiment, we also compare with these approaches to demonstrate the advantages of our model.

Our approach. In this paper, to identify the top- k vulnerable nodes, we model the problem with an uncertain graph, and infer the default probability of a node following the possible world semantics, which has been widely used to capture the contagious phenomenon in real networks [11, 12, 13, 14]. In particular, we utilize an uncertain graph with two types of probabilities to model the occurrence and prorogation of the default risks in the network. Note that, we focus on identifying vulnerable nodes for a given network, while the self-risk probabilities and diffusion probabilities can be obtained based on the existing works (e.g., [15, 10]).

Specifically, Figures 3(a) and 3(e) illustrate the structure of a toy uncertain graph with 5 nodes and 6 edges, as well as the associated self-risk probabilities and diffusion probabilities. Given the probabilistic graph \mathcal{G} , we may derive the *default probability* of a node following the possible world semantics, where each possible world (i.e., *instance graph* in this paper) corresponds to a subgraph (i.e., possible occurrence) of \mathcal{G} . Figures 3(b)-(d) are three example possible worlds of Figure 3(a). In each possible world, a node exits if it defaults, and an edge (A, B) appears if the default of A indeed leads to the default of B . Taking node E as an example, it may default because of (i) itself, which is represented by a shaded node as shown in Figure 3(b), or because of (ii) the contagion damage initiated by other nodes as shown in Figures 3(c)-(d). In Section 2, we will formally introduce how to derive the default probabilities of nodes.

In this paper, we show that the problem of calculating the default probability of a node alone is #P-hard, not mentioning the top- k vulnerable nodes identification problem. A straightforward solution for the top- k vulnerable nodes computation is to enumerate all possible worlds and then aggregate the results in each possible world. However, it is computational prohibitive, since the number of possible worlds of an uncertain graph may be up to 2^{n+m} , where n and m are the number of nodes and edges in the graph, respectively. In this paper, we first show that we can identify the top- k vulnerable nodes by using a limited number of sampled instance graphs with tight theoretical guarantee. To reduce the sample size required and speedup the computation, lower/upper bounds based pruning strategies and advanced sampling method are developed. In addition, to further accelerate the computation, a bottom- k sketch based method is proposed. To verify the performance in real scenarios, we integrate the proposed techniques with our current loan risk control system, which is deployed in the collaborated bank.

Contributions. The principle contributions of this paper are summarized as follows.

- We advocate the problem of top- k vulnerable nodes detection in uncertain graphs, which is essential in real-world applications.
- Due to the hardness of the problem, a sampling based method is developed with tight theoretical guarantee.
- We develop effective lower and upper bound techniques to prune the searching space and reduce the sample size required. Advanced sampling method is designed to speed up the computation with rigorous theoretical analysis.
- A bottom- k sketch based approach is further proposed, which can greatly speedup the computation with competitive results.
- We conduct extensive experiments to evaluate the efficiency and effectiveness of our proposed algorithms on 3 real financial networks and 5 benchmark networks.

Table 1: Summary of notations

Notation	Definition
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	uncertain graph
\mathcal{V}/\mathcal{E}	node/edge set of \mathcal{G}
$p_s(v)$	the self-risk probability of v
$p(v_i v_j)$	the diffusion probability
$p(v)$	the default probability of v
$h(x)$	a truly random hash function
$\mathcal{L}(A, k)$	the k -th smallest hash value of the set A
bk	the parameter k in the bottom- k sketch
$N(v)$	the set of in-neighbors of node v
\mathcal{A}	an approximation algorithm for the problem
\mathcal{R}	the set of k nodes returned by \mathcal{A}
P^k	the default probability of rank k -th nodes
(ϵ, δ)	the parameters in (ϵ, δ) -approximation
t	the sample size
$p_l(v), p_u(v)$	the lower and upper bound of $p(v)$
T_l, T_u	the k -th largest value of $p_l(v)$ and $p_u(v)$
\mathcal{B}	the set of candidates
\mathcal{G}^t	graph by reversing the direction of edges in \mathcal{G}

- To further verify the advantages of our models in real scenarios, the proposed techniques are integrated into our current loan risk control system, which is deployed in the collaborated bank. Through the case study on real-life financial environment, it verifies that our proposed model can significantly improve the accuracy for high-risky enterprises prediction.

Roadmap. The rest of the paper is organized as follows. Section 2 describes the problem studied and the related techniques used in the paper. Section 3 shows the basic sampling-based method and our optimized algorithms. We report the experiment results in Section 4. Section 5 introduces the system implementation details and case study. We present the related work in Section 6 and conclude the paper in Section 7.

2 Preliminaries

In this section, we first we present some key concepts and formally define the problem. Then, we introduce the related techniques used. Table 1 summarizes the notations frequently used throughout this paper.

2.1 Problem Definition

We consider an uncertain graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a directed graph, where \mathcal{V} is the set of nodes and \mathcal{E} denotes the set of edges. $n = |\mathcal{V}|$ (resp. $m = |\mathcal{E}|$) is the number of nodes (resp. edges) in \mathcal{G} . Each node $v \in \mathcal{V}$ is associated with a **self-risk probability** $p_s(v)$, which denotes the default probability of v caused by self-factors. Each edge $(u, v) \in \mathcal{E}$ is associated with a **diffusion probability** $p(v|u)$, which denotes the probability of v 's default caused by u 's default. In this paper, we assume the self-risk probabilities and diffusion probabilities are already available. These probabilities can be derived based the previous studies (e.g., [10]).

For simplicity, when there is no ambiguity, we use uncertain graph, graph and network interchangeably. In this paper, we derive the default probability of a node by considering both self-risk probability and diffusion probability, which is defined as follows.

Definition 1 (Default Probability). *Given an uncertain graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, for each node $v \in \mathcal{V}$, its default probability, denoted by $p(v)$, is obtained by considering both self-risks probability and diffusion probability. $p(v)$ can be computed recursively as follows.*

$$p(v) = 1 - (1 - p_s(v)) \left(\prod_{\text{all } x \in N(v)} (1 - p(v|x)p(x)) \right) \quad (1)$$

where $N(v)$ is the set of in-neighbors of v .

It is easy to verify that the equation above is equal to aggregate the probability over all the possible worlds, i.e.,

$$p(v) = \sum_{W \in \mathcal{W}} p(W) \times I_W(v)$$

where \mathcal{W} is the set of all possible worlds, $p(W)$ is the probability of a possible world W , and $I_W(v)$ is an indicator function denoting if v defaults in W or not.

Example 1. Reconsider the graph in Figure 3. Suppose the self-risk probabilities and diffusion probabilities are all 0.2 for each node and edge. Then, we have $p(A) = 0.2$ and $p(B) = 1 - (1 - p_s(B))(1 - 0.2 \times p(A)) = 0.232$.

Problem statement. Given an uncertain graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we aim to develop efficient algorithms to identify the set \mathcal{R} of top- k vulnerable nodes, i.e., the k nodes with the highest default probability.

Theorem 1. It is #P-hard to compute the default probability.

Proof. We show the hardness of the problem by considering a simple case, where the self-risk probability $p_s(v)$ equals 1 for node v , and $p_s(u)$ equals 0 for $u \in \mathcal{V} \setminus v$. Therefore, for the node $u \in \mathcal{V} \setminus v$, the default probability $p(u)$ is only caused by the default of node v . Then, the default probability of $p(u)$ equals the reliability from v to u , which is #P-hard to compute [16]. Thus, it is #P-hard to compute the default probability. The theorem is correct. \square

2.2 Bottom- k Sketch

In this section, we briefly introduce the bottom- k sketch [17], which is used in our BSRBK framework to obtain the statistics information for early stopping condition. Bottom- k sketch is designed for estimating the number of distinct values in a multiset. Given a multiset $A = \{v_1, v_2, \dots, v_n\}$ and a truly random hash function h , each distinct value v_i in the set A is hashed to $(0, 1)$ and $h(v_i) \neq h(v_j)$ for $i \neq j$. The bottom- k sketch consists of the k smallest hash values, i.e., $\mathcal{L}(A) = \{h(v_i) | h(v_i) \leq \mathcal{L}(A, k) \wedge v_i \in A\}$, where $\mathcal{L}(A, k)$ is the k -th smallest hash value. So the number of distinct value can be estimated with $\frac{k-1}{\mathcal{L}(A, k)}$. The estimation can converge fast with the increase of k , where the expected relative error is $\sqrt{2/\pi(k-2)}$ and the coefficient variation is no more than $1/\sqrt{k-2}$. To distinguish from the k in the top- k problem, hereafter in this paper, we use bk to denote the parameter k in the bottom- k sketch.

3 Solutions

In this section, we first present a basic sampling method and analyze the sample size required. Then, we introduce the optimized approaches to accelerate the processing.

3.1 Basic Sampling Approach

Due to the hardness of computing the default probability, in this section, we propose a sampling based method. In order to bound the worst case performance, rigorous theoretical analysis is conducted about the required sample size.

Sampling framework. To compute the default probability, we can enumerate all the possible worlds and aggregate the results. However, the possible world space is usually very large. In previous research, sampling based methods are widely adopted for this case. That is, we randomly select a set of possible worlds and take the average value as the estimated default probability. By carefully choosing the sample size, we can return a result with performance guarantee.

Algorithm 1 shows the details of the basic sampling based method. The input is a given graph, where each node/edge is associated with a self-risk/diffusion probability. In each iteration, we generate a random number for each node to determine if it defaults by itself or not (Lines 4-7). Then, we conduct a breath first search from these nodes, i.e., $h_v = 1$, to locate the nodes that will be influenced by them in the current simulation. For each encountered edge, we generate a random number to decide if the propagation will continue or not. For each node, the number of default times is accumulated in Lines 21. The final default probability is calculated by taking an average over the accumulated value v^c . Finally, the algorithm returns k results with the largest estimated value.

Example 2. Reconsider the example in Figure 3. Suppose Figures 3(b)-3(d) are the three sampled graphs, and nodes E, D, A are the ones default by itself. Assuming $k = 2$, then, nodes E and D are returned results with the largest estimated default probabilities.

Sample size analysis. For sampling based methods, a critical problem is to determine the sample size required in order to bound the quality of returned result. In this section, we conduct rigorous theoretical analysis about the sample size required. Specifically, we say an algorithm \mathcal{A} is (ϵ, δ) -approximation if the following conditions hold.

Algorithm 1: Basic Sampling Approach

Input : \mathcal{G} : a given graph, k : a positive integer, t : sample size

Output : \mathcal{R} : collection of top- k vulnerable nodes

```
1   $u_v := 0$  for all nodes  $v \in \mathcal{V}$  ;
2  for  $i$  in 1 to  $t$  do
3     $h_v := 0$  for all nodes  $v \in \mathcal{V}$  ;
4    for each  $v \in \mathcal{V}$  do
5      generate a random number  $r_v$  in  $[0, 1]$ ;
6      if  $r_v \leq p_s(v)$  then
7         $h_v := 1$ ;
8     $\mathcal{Q} \leftarrow \{v | h_v = 1\}$  ;
9    mark  $\mathcal{V} \setminus \mathcal{Q}$  as unvisited ;
10   while  $\mathcal{Q} \neq \emptyset$  do
11      $v_q \leftarrow \mathcal{Q}.pop()$  ;
12     for each  $v_a \in N(v_q)$  do
13       if  $v_a$  is unvisited then
14         generate a random number  $r_e$  in  $[0, 1]$  ;
15         if  $r_e > p(v_a | v_q)$  then
16           continue ;
17          $h_{v_a} := 1$  ;
18         mark  $v_a$  as visited ;
19         push  $v_a$  to  $\mathcal{Q}$  ;
20   for each  $v \in \mathcal{V}$  do
21      $v^c := v^c + h_v$  ;
22    $\mathcal{R} \leftarrow$  the top- $k$  nodes ordered by  $p_v = v^c / t$  ;
23   return  $\mathcal{R}$ 
```

Definition 2 ((ϵ, δ) -approximation). *Given an approximation algorithm \mathcal{A} for the top- k problem studied, let \mathcal{R} be the set of k nodes returned by \mathcal{A} . P^k is the default probability of the k -th node in the ground truth order. Given $\epsilon, \delta \in (0, 1)$, we say \mathcal{A} is (ϵ, δ) -approximation if \mathcal{R} fulfills the following conditions with at least $1 - \delta$ probability.*

- 1) For $v \in \mathcal{R}$, $p(v) \geq P^k - \epsilon$;
- 2) For $v \notin \mathcal{R}$, $p(v) < P^k + \epsilon$.

If \mathcal{A} is a (ϵ, δ) -approximation algorithm for the vulnerable node detection problem, it means that with high probability (i) the default probabilities of returned nodes are at least $P^k - \epsilon$; (ii) for the nodes not in \mathcal{R} , the default probabilities are at most $P^k + \epsilon$.

Theorem 2 (Hoeffding Inequality). *Given a sample size t and $\epsilon > 0$, let p_v be the unbiased estimation of $p(v)$, where $p_v = \frac{1}{t} \sum_{i=1}^t p_v^i$ and $p_v^i \in [a^i, b^i]$. Then we have*

$$Pr[p_v - p(v) \geq \epsilon] \leq \exp\left(-\frac{2t^2\epsilon^2}{\sum_{i=1}^t (a^i - b^i)^2}\right) \quad (2)$$

Based on the Hoeffding inequality, we have following theorem hold.

Theorem 3. *Given the sample size t , $\epsilon > 0$ and two nodes $u, v \in \mathcal{V}$, if $p(v) - p(u) \geq \epsilon$, then*

$$Pr[p_u - p_v > 0] \leq \exp(-t\epsilon^2/2)$$

Proof. We have

$$\begin{aligned}
& Pr[p_u - p_v > 0] \\
& \leq Pr[p_u - p_v \geq p(u) - p(v) - \epsilon] \\
& = Pr[p_u - p_v - (p(u) - p(v)) \geq \epsilon] \\
& \leq \exp\left(-\frac{2t^2\epsilon^2}{\sum_{i=1}^t 2^2}\right) \\
& = \exp(-t\epsilon^2/2)
\end{aligned}$$

The last two steps consider $p_u - p_v$ as the estimator of $p(u) - p(v)$. In addition, $p_u^i - p_v^i \in [-1, 1]$. Then, we can feed into the Hoeffding inequality and obtain the final result. \square

As shown in Theorem 4, Algorithm 1 is a (ϵ, δ) -approximation algorithm if the sample size is no less than Equation 3.

Theorem 4. *Algorithm 1 is (ϵ, δ) -approximation if the sample size is no less than*

$$t = \frac{2}{\epsilon^2} \ln \frac{k(n-k)}{\delta} \quad (3)$$

where n is the number of nodes, i.e., $|\mathcal{V}|$.

Proof. Suppose we sort the nodes based on their real default probabilities, i.e., $\{v_1, v_2, \dots, v_n\}$. Then we show the two conditions in (ϵ, δ) -approximation hold if we have $p_{v_i} - p_{v_j} > 0$ with $p(v_i) - p(v_j) \geq \epsilon$ for $1 \leq i \leq k < j \leq n$.

- For a node $v \in \mathcal{R}$, if $p(v) < P^k - \epsilon < p(v_i) - \epsilon$, it means $p_{v_i} - p_v > 0$ for $i \in [1, k]$. Therefore, v will not be selected into \mathcal{R} , which is contradict to the assumption. Thus, the first condition holds.
- For $v \notin \mathcal{R}$, if v does not belong to the top- k result, the second condition holds naturally. Otherwise, there must be a node u that does not belong to the top- k result being selected into \mathcal{R} . If $p(v) \geq P^k + \epsilon \geq p(u) + \epsilon$, it means $p_v - p_u > 0$. Therefore, v should also be selected into the top- k , which is contradict to the assumption. Thus, the second condition holds.

Theorem 3 shows the theoretical result of bounding the order of a pair of nodes. Since $1 \leq i \leq k < j \leq n$, we need to bound the order of $k(n-k)$ pairs of nodes. By applying union bound and Theorem 3, we have

$$\begin{aligned}
\delta & = k(n-k) \exp(-t\epsilon^2/2) \\
\Rightarrow t & = \frac{2}{\epsilon^2} \ln \frac{k(n-k)}{\delta}
\end{aligned}$$

Therefore, the theorem is correct. \square

3.2 Optimized Sampling Approach

Based on Theorem 4, Algorithm 1 can return a result with tight theoretical guarantee. However, it still suffers from some drawbacks, which make it hard to scale for large graphs. Firstly, to bound the quality of returned results, we need to bound the order of $k(n-k)$ node pairs. The node size n can be treated as the candidate size, which is usually large in real networks. Therefore, if we can reduce the size of n (i.e., reduce candidate space) and k (i.e., verify some nodes without estimation), then the sample size can be reduced significantly. Secondly, in each sampled possible world, we only need to determine whether the candidate node can be influenced or not, i.e., compute h_v . If the candidate space can be greatly reduced, the previous sampling method may explore a lot of unnecessary space.

According to the intuitions above, in this section, novel methods are developed to derive the lower and upper bounds of the default probability, which are used to reduce the candidate space. In addition, a reverse sampling framework is proposed in order to reduce the searching cost.

Candidate size reduction. To compute the lower and upper bounds of the default probability, we utilize the equation in default probability definition, i.e., Equation 1. The idea is that the default probability for each node is in $[p_s(v), 1]$ if no further information is given. By treating each node's default probability as $p_s(v)$ and 1, we can aggregate the probability over its neighbors to shrink the interval based on Equation 1. Then, with the newly derived lower and upper bounds for neighbor nodes, we can further aggregate the information and update the bounds. The details of deriving

Algorithm 2: Lower Bound Algorithm

Input : \mathcal{G} : a given graph, z : the order of bound**Output** : $p_l(v)$: lower bound of default probability

```
1 for  $i$  in 1 to  $z$  do
2   if  $i = 1$  then
3      $p(v) := p_s(v)$  for each  $v \in \mathcal{V}$ ;
4     continue;
5   for each  $v$  in  $\mathcal{V}$  do
6     calculate  $p(v)$  by Equation 1 if its in-neighbors' default probabilities have been updated in the previous iteration;
7  $p_l(v) := p(v)$  for all nodes  $v \in \mathcal{V}$ ;
8 return  $p_l(v)$  for  $v \in \mathcal{V}$ 
```

Algorithm 3: Upper Bound Algorithm

Input : \mathcal{G} : a given graph, z : the order of bound**Output** : $p_u(v)$: upper bound of default probability

```
1 for  $i$  in 1 to  $z$  do
2   for each vertex  $v$  in  $\mathcal{V}$  do
3     if  $i = 1$  then
4       calculate  $p(v)$  by treating its in-neighbors' default probabilities as 1;
5     else
6       calculate  $p(v)$  if its in-neighbors' default probabilities have been updated in the previous iteration;
7  $p_u(v) := p(v)$  for all nodes  $v \in \mathcal{V}$ ;
8 return  $p_u(v)$  for  $v \in \mathcal{V}$ 
```

lower and upper bounds are shown in Algorithms 2 and 3. The algorithms iteratively use the lower and upper bound derived in the previous iteration as the current default probability. The order of bound denotes the number of iterations conducted. In each iteration, we only update the bounds of v 's default probability if there is any change for the default probability of v 's in-neighbors. Note that, there is a slight difference in the first iteration for the two algorithms, since by using 1 as the default probability will contribute nothing for the pruning. It is easy to verify that larger order will lead to tighter bounds. Users can make a trade-off between the efficiency and the tightness of bounds.

Given the lower bound and upper bound derived, we can filter some unpromising candidates and verify some candidates with large probability. Lemma 1 shows the pruning rules to verify and filter the candidate space.

Lemma 1. *Given the upper and lower bounds derived for each node, let T_l and T_u be the k -th largest value in $p_l(v)$ and $p_u(v)$, respectively. Then, we have*

- 1) For $u \in \mathcal{V}$, u must be in the top- k if $p_l(u) \geq T_u$.
- 2) For $u \in \mathcal{V}$, u must not be in the top- k if $p_u(u) < T_l$.

Proof. For the first case, suppose a node u with $p_l(u) \geq T_u$ but not being selected in the top- k results, which means a node must have default probability of at least $p_l(u)$ to be selected into the top- k result. Since T_u is the k -th largest value in $p_u(v)$, it means there will be no more than k nodes that satisfy the condition. Therefore, the first case holds. For the second case, since T_l is the k -th largest value of $p_l(v)$, which means P^k must be at least T_l . Note that, P^k is default probability of the ranked k -th node in the the ground truth order. Therefore, the second case holds. \square

Based on Lemma 1, Algorithm 4 shows the details of reducing candidate space. The algorithm takes the derived lower and upper bounds as input and outputs the candidate nodes \mathcal{B} and the number k' of verified nodes. The verified k' nodes will be put into the result set directly. Note that, if we can verify k' nodes based on the first pruning rule, then we only need to find top- $(k - k')$ nodes from the candidate set \mathcal{B} . In this case, we reduce both the value k and n of Equation 3 to $k - k'$ and $|\mathcal{B}|$, respectively.

Reverse sampling approach. Based on Algorithm 4, we can greatly reduce the candidate space, which performance is verified in our experiments on real-world datasets. In the basic sampling method, it aims to estimate the default probability for each node. Here, we only need to compute the probability for the candidate nodes. Especially, when the candidate size is small, the previous sampling method will explore a lot of unnecessary space.

Algorithm 4: Candidate Reduction

Input : $p_u(v)/p_l(v)$: upper and lower bound for each node, k : a positive integer**Output** : \mathcal{B} : candidates selected, k' : the number of nodes verified

```
1  $T_l \leftarrow$  the  $k$ -th largest value in  $p_l(v)$  ;
2  $T_u \leftarrow$  the  $k$ -th largest value in  $p_u(v)$  ;
3  $\mathcal{B} := \emptyset$ ;  $k' = 0$  ;
4 for each  $v$  in  $\mathcal{V}$  do
5   if  $p_l(v) \geq T_u$  then
6      $k' ++$ ;
7     insert  $v$  into the result set;
8     continue;
9   if  $p_u(v) \geq T_l$  then
10    push  $v$  into  $\mathcal{B}$  ;
11 return  $\mathcal{B}$  and  $k'$ 
```

Algorithm 5: Reverse Sampling Algorithm

Input : \mathcal{G}^t : a given graph by reverse the direction each edge, \mathcal{B} : candidate nodes**Output** : h_v : for each node $v \in \mathcal{B}$ in one sample

```
1  $h_v := 0$  for all nodes  $v \in \mathcal{V}$  ;
2 for each node  $v$  in  $\mathcal{B}$  do
3   mark all nodes as unvisited ;
4    $\mathcal{Q} \leftarrow \{v\}$  ;
5   while  $\mathcal{Q} \neq \emptyset$  do
6      $u = \mathcal{Q}.pop()$  ;
7     if  $h_u = 1$  then
8        $h_v := 1$  and break ;
9     if  $u$  is unchecked then
10      generate a random number  $r_u$  in  $[0, 1]$  ;
11      mark  $u$  as checked ;
12      if  $r_u \leq p_s(u)$  then
13         $h_u := 1, h_v := 1$  and break ;
14      for each  $u' \in N(u)$  do
15        if  $(u', u)$  is unchecked then
16          generate a random number to mark  $(u', u)$  as survived or not ;
17      mark  $u$  as visited ;
18      for each  $u' \in N(u)$  do
19        if  $u'$  is unvisited and  $(u', u)$  is survived then
20          push  $u'$  into  $\mathcal{Q}$  ;
21 return  $h_v$  for nodes in  $\mathcal{B}$ 
```

Intuitively, given a sampled possible world, for each candidate node, we only need to verify if it can be reached by a node with $h_v = 1$. Therefore, we can conduct a reverse traversal from the candidate nodes to see if it can meet the criteria. The details are shown in Algorithm 5, where \mathcal{G}^t is the graph by reversing the direction of each edge in \mathcal{G} . Note that, our reverse sampling method is different from the reverse sampling framework used in influence maximization problem [18].

The inputs are the graph \mathcal{G}^t and candidate set \mathcal{B} . After processing a sample, it returns h_v for each node v in \mathcal{B} . At first, we set $h_v = 0$ for all the nodes. Then we conduct a breath first search from each node in the candidate set. For each encountered node and edge, we mark it as *checked* and store the corresponding information (e.g., *survived* and h_v) in order to avoid generating random numbers for the same node/edge multiple times. The BFS terminates if it encounters a node h_v with $h_v = 1$ or there is no more node to be explored (Lines 6-8). If it encounters a node with $h_v = 1$, it means the candidate node is influenced, and vice versa. Through this way, we can greatly reduce the computation cost by filtering unnecessary searching space. Given sample size t , we repeat the process t times and accumulate the h_v value to estimate the default probability.

Reverse sampling based method. By integrating the pruning strategies and the reverse sampling technique, we have the reverse sampling based algorithm. According to Theorem 5, the approach is (ϵ, δ) -approximation if the sample size fulfills Equation 4.

Theorem 5. *The reverse sampling based algorithm is (ϵ, δ) -approximation if the sample size is larger or equal than*

$$t = \frac{2}{\epsilon^2} \ln \frac{(k - k')(|\mathcal{B}| - k + k')}{\delta} \quad (4)$$

Note that, we use the reverse sampling method here because the candidate space is reduced. In addition, it only accelerate the computation of influenced nodes in a sampled possible world. Based on the developed pruning techniques, we can verify some results and filter some candidates. To bound the quality of returned results, we need to bound the order of $(k - k')(|\mathcal{B}| - k + k')$ pairs.

3.3 Bottom-k Based Approach

Based on the lower and upper bounds derived, we can reduce the candidate space. In addition, by using the reverse sampling technique, we can reduce the cost of exploring samples. The reverse sampling based algorithm can return a result with tight theoretical guarantee, which reduces the sample size from Equation 3 to Equation 4. However, in many real cases, the sample size and computation cost is still large. Intuitively, we only need sufficient samples to obtain a competitive result. In this section, we derive a method by using the bottom- k technique, which can greatly accelerate the procedure with competitive top- k results. We first present the idea of finding the top-1 result. Then, we extend the idea for the top- k scenario.

Finding the top-1 result. In the reverse sampling approach, when we process the samples one by one. We can terminate the processing, if there is a node that has sufficient statistic. In this paper, we use bottom- k sketch to serve this role. The idea is that, we first apply the lower and upper bound technique to obtain k' and \mathcal{B} . Let t be sample size computed by using Equation 4. We assign each sample an id and generate a random hash value in $(0, 1)$ for each of them. Since we does not materialize the samples, the time complex of generating hash value is only $O(t)$. We sort the samples in ascending order based on the hash value, and materialize the samples accordingly by using the reverse sampling framework. For each node v in the candidate set, we record a accumulated value v^c . Based on Theorem 6, the node whose v^c reaches bk first is the top-1 result. bk is the threshold selected.

Theorem 6. *The node selected by using the above procedure is the top-1 node.*

Proof. Suppose node u is the first node that reaches the criteria and the hash value of its bk -th encountered sample is $h^{bk}(u)$. According to the property of bottom- k sketch, we can estimate the default probability $p(u)$ with $\frac{bk-1}{h^{bk}(u)t}$. If v is the second node that reaches the criteria. We must have $h^{bk}(v) > h^{bk}(u)$. Therefore, the corresponding estimated value is smaller that of u . The theorem is correct. \square

Here, we use bk to measure if the statistic is sufficient or not. Even though the bottom- k based method does not offer tight theoretical guarantee as the previous approaches. Through our experimental evaluation, the bottom- k based method shows great advantage compared with the others.

Example 3. *Reconsider the graph in Figure 3 with $k = 1$ and $bk = 2$. Suppose nodes D and E are the candidates in \mathcal{B} . Then, we only need to reverse the graph and conduct the traversal from D and E . For the simplicity, we do not present the reverse graph here. Figure 3(b) is the first reverse sample. That is, E is default by itself, and D does not meet any default nodes when back-traversing from D . Therefore, the counter of E is set as 1. Figure 3(c) is the second sample. Then, the counter of E becomes 2, which meets the bottom- k criterion. Thus, D is the top- k result returned.*

Finding the top- k results. To find the top- k , we follow similar manner as that in finding the top-1 result. By extending Theorem 6, we can stop exploring the samples when there are $k - k'$ nodes with sufficient statistic. That is, we continue visiting the samples until there are $k - k'$ nodes with counters reaching bk . Note that, there may be case when the stop condition cannot be met after all the samples are processed. Then, the algorithm turns to the reverse based sampling method, and we just return the $k - k'$ nodes with the largest estimated value. While, according to the experiments over real-world datasets, the algorithm can coverage quickly with bk .

Complexity analysis. In this paper, there are two types of samples involved, i.e., basic sample (Algorithm 1) and reverse sample (Algorithm 5). We use t_b and t_r to denote the cost of generating a basic sample and a reverse sample, respectively. To obtain a sample, in the worst case, we need to traverse the whole graph once for both sampling approaches, which cost is $O(m + n)$. Therefore, for the basic sampling approach (i.e., Algorithm 1), the time

Table 2: Details of experimental datasets

Datasets	# Nodes	# Edges	Avg Deg.	Max Deg.
Bitcoin	3,783	24,186	6.39	888
Facebook	4,039	88,234	21.85	1,045
Wiki	7,115	103,689	14.57	1,167
P2P	62,586	147,892	2.36	95
Citation	2,617	2,985	1.14	44
Interbank	125	249	1.99	47
Guarantee	31,309	35,987	1.15	14,362
Fraud	14,242	236,706	16.62	85,074

complexity is $O((m+n)\frac{2}{\epsilon^2}\ln\frac{k(n-k)}{\delta}+n\log n)$, where ϵ and δ are the input parameters. For the reverse sampling based method, the bound computation and candidate reduction phase cost $O(z(m+n))$ and $O(n\log n)$, respectively. Thus, the complexity is $O(t_r\frac{2}{\epsilon^2}\ln\frac{(k-k')(|\mathcal{B}|-k+k')}{\delta}+z(m+n)+2n\log n)$. The time complexity of the bottom- k based method is the same as that of the reverse sampling based approach. This is because, we need to explore all the samples in the worst case, but it is usually much faster than others in practice.

4 Experiment

In this section, we conduct extensive experiments to evaluate the effectiveness and efficiency of our proposed methods.

4.1 Experiment Setup

Datasets. We conduct the experiments on 3 real-world financial datasets, i.e., Interbank¹, Fraud and Guarantee and 5 public benchmark datasets with drastically varying sizes and characteristics. Fraud and Guarantee are our contributed dataset. The details about the 3 real-world financial datasets are shown as follows.

- **Interbank.** Interbank networks is generated by the maximum-entropy (ME) approach [19], in which each node represents a bank and edge corresponds to an interbank loan from the lender bank to the borrow bank.
- **Fraud.** Credit card fraud networks with 19, 240 nodes and 34, 892 edges is constructed based on credit card fraud transactions of a major commercial bank. Each edge represents a trade between the consumer and merchant.
- **Guarantee.** The guaranteed loans network is from a major commercial bank spanning 4 years. The names of the customers in the records are encrypted and replaced by IDs. We can access the guarantee relationships, which denotes an edge between the guarantor to borrower. Besides, in case studies, we also get the basic profile information such as the enterprise scale, and loan information such as the loan credit.

Besides the real-world financial datasets, we also employ 5 benchmark datasets, which are public available. We download Citation from network repository². The others are downloaded from SNAP³. The statistic details of datasets are shown in Table 2.

Algorithms. We evaluate the following algorithms to demonstrate the performance of following algorithms.

- N (*Naive*). Algorithm 1 with fixed sample size.
- SN (*Naive+Sample*). Algorithm 1 with the sample size calculated by Equation 3.
- SR (*Sample+Reverse*). Algorithm that uses reverse sampling method with candidate set derived with second rule of Lemma 1.
- BSR (*Bound+Sample+Reverse*). Optimized sampling method by integrating reverse sampling and bounds filtering techniques with the sample size calculated by Equation 4.

¹<https://github.com/carloscinelli/NetworkRiskMeasures>

²<http://networkrepository.com/>

³<https://snap.stanford.edu/data/>

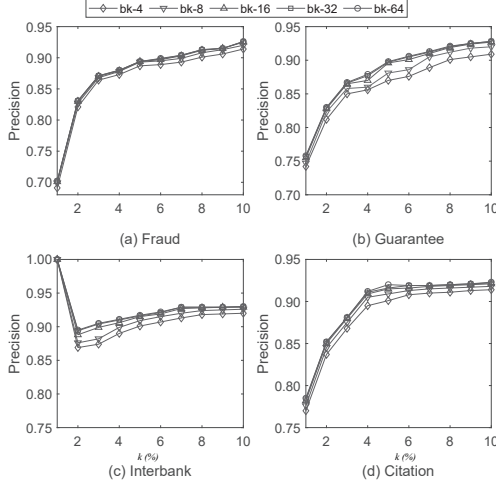


Figure 4: Parameter bk tuning for bottom- k based method

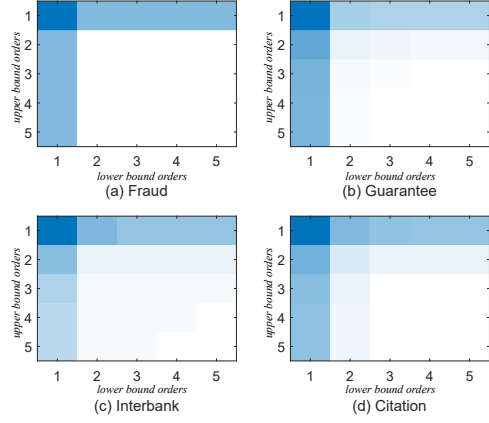


Figure 5: Parameter tuning for the order of bounds

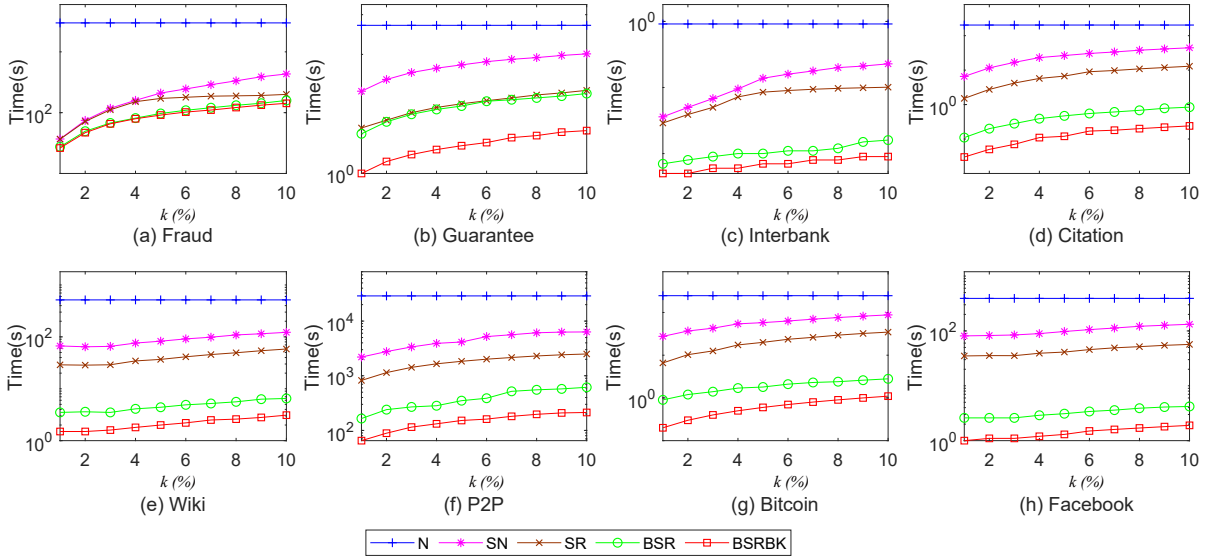


Figure 6: Efficiency evaluation

- BSRBK (*Bound+Sample+Reverse+Bottom-k*). Bottom- k based method by integrating reverse sampling and bounds filtering techniques.

Parameters and workload. To evaluate the effectiveness of proposed techniques, the precision is reported. For the ground truth, we use 20000 sampled possible worlds to obtain the results, which setting is commonly used in related research [14, 6, 5]. For Fraud and Guarantee datasets, the self-risk and diffusion probability are obtained in our previous research [20, 15]. For the other datasets, the probability is randomly selected from $[0, 1]$. For parameter k , we vary it from $1\%|\mathcal{V}|$ to $10\%|\mathcal{V}|$, where $|\mathcal{V}|$ is the corresponding graph node size. We set $\epsilon = 0.3$ and $\delta = 0.1$ for computing the sample size.

In the experiment, we run all the methods by a server with two Intel E5-2680 CPU, 512GB memory and CentOS v7.2 operation system. In the system implementation, we develop the web backend server by Spring Cloud and frontend with JavaScript D3.js library. In case studies, CNN-max, crDNN and HGAR are experimented on a GPU server with two pieces of Tesla-P100 and implemented by Tensorflow.

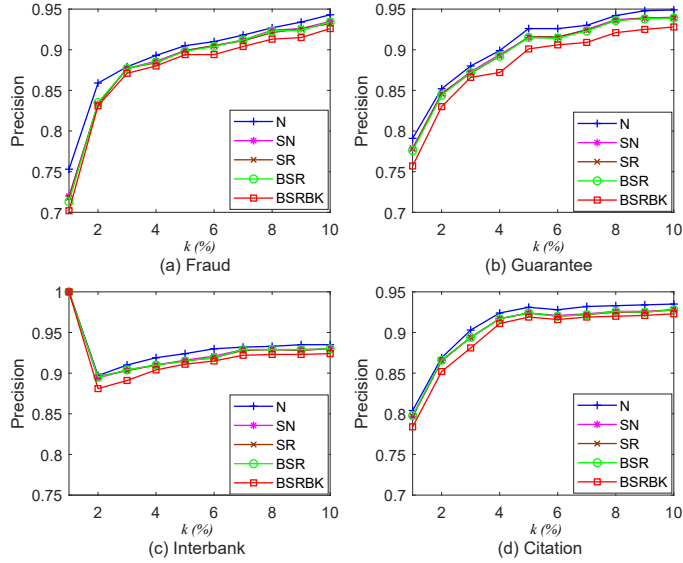


Figure 7: Effectiveness evaluation

4.2 Parameter Tuning

In this section, we tune the parameters bk and the order of bounds on 4 datasets, i.e., Citation, Interbank, Fraud and Guarantee.

Tuning the parameter bk . As analyzed in the paper, the precision of BSRBK should converge rapidly with the increase of bk . We vary bk from 4 to 64. The results are shown in Figure 4. Note that $bk-X$ means bk is set to X . With the increase of bk , the algorithm converges quickly for all the datasets. When bk reaches 8, the drop of performance already becomes less significant. Thus, in the following experiments, we set bk to 16.

Tuning the order of bounds. Since the tightness of lower and upper bounds may greatly affect the sample size and computation cost, we conduct the experiments to tune the order of bounds. We vary the order of bounds from 1 to 5 and set k as 5% of the number of nodes. The candidate size is reported. Figure 5 visualizes the result with heatmaps. The lighter the color is, the less number of candidates will be. As we can see, the candidate size decreases rapidly at the beginning, and reach steady when the order reaches 2 for most cases. Therefore, we set the order of upper and lower bounds to 2 for the following experiments.

4.3 Efficiency Evaluation

To demonstrate the efficiency of proposed techniques, we conduct experiments on all the datasets and report the response time. The results are shown in Figure 6. In all methods, the computation time gradually increases along with k except for the naive approach N, because N uses a large fixed sample size. For the other methods, the sample size may change when k increases. As we can observe, algorithm N is the most time-consuming method, and the algorithm runs faster when more accelerating techniques involved. SR is better than SN because the reverse sampling technique and candidate set derived can greatly reduce the sampling cost. BSR is better than SR, since we can reduce the candidate space and sample size by using the lower and upper bounds derived. BSRBK is better than BSR because of the novel stop condition used. BSRBK always outperforms the others and achieves up to 100x acceleration. These observations strongly proves the advantage of proposed techniques.

4.4 Effectiveness Evaluation

To evaluate the effectiveness of proposed methods, the precision is reported by varying k from $1\%|\mathcal{V}|$ to $10\%|\mathcal{V}|$. The results are shown in Figure 7. Generally, the precision of the 5 methods is very close to each other, and the largest gap between the naive method N and BSRBK is only 3%. Compared with the speedup in efficiency, the precision difference is much less noticeable. The naive method N is slightly better than the other methods, because it has used more samples. SN, SR and BSR report almost the same result, because they obtain the same theoretical guarantee. It should be noted that for the Interbank dataset, $1\%|\mathcal{V}| = 1$ and all methods successfully detect that node. Therefore, the precision is 1 as

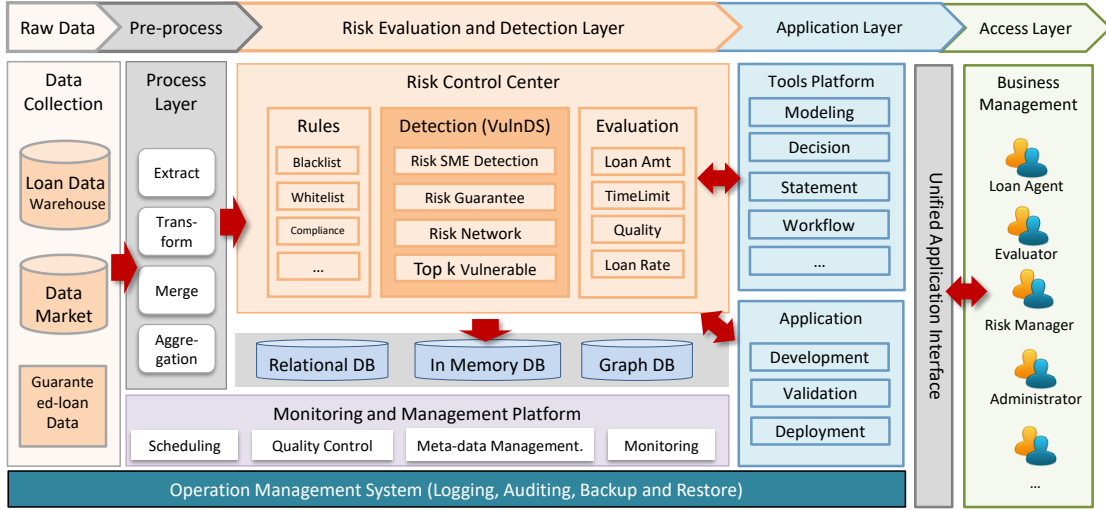


Figure 8: System architecture of VulnDS in a loan management system

shown in Figure 7(c). As observed, the experiment results prove that BSRBK could achieve significant performance acceleration while keeping a tolerable precision reduction.

5 Implementation and Case Study

In this section, We present a system, named VulnDS, by integrating the proposed techniques with our current loan risk control system. The control system is deployed in our collaborated bank, which can evaluate our methods in real scenarios. We first present the overall architecture for VulnDS, and then describe the details of system implementation. Finally, we report the interface, observation and case study after system deployment.

5.1 System Implementation and Deployment

Architecture Overview. Figure 8 shows the architecture overview of the VulnDS in a loan management system. We collect origin data from three upstream: loan data warehouse, data market and external loan data. In the pre-processing layer, raw records are extracted, merged and aggregated for risk control. We employ the in-memory database to store the frequent queried data, and graph database to preserve networked relationships, as well as rational DB for conventional tables. We utilize a monitoring platform for scheduling submitted tasks from the pre-processing and risk control module. The risk assessment results are consumed by the tools and application platform, which is the main scenario to control loan risks. Different roles of business users access the system from a unified application interface.

The risk control center consists of three main parts: the rule engine, vulnerable detection system and evaluation module. Rule engine mainly includes loan blacklist, white list and compliance rules. If a loan passes the rule check, it will be then processed by our proposed vulnerable detection system. VulnDS assess the self-risk of SME, the risk of guarantee relationships, and detect the top- k vulnerable nodes by our methods. Evaluation module leverage the output of VulnDS to quantify the loan grant amount, time limit and interest ratio, etc. Once the bank issue a loan, post-loan process are activated immediately. All three steps in the risk control center will be employed to evaluate all issued loans regularly. In our implementation, we detect all loans monthly by the proposed VulnDS in a risk control center.

Implementation Details. Figure 9 shows an overview of the data association, which is extracted by the pre-processing layer. We employ the internal black and white lists from our collaborated bank. The rules are mainly under the compliance of the new Basel protocol[21]. In vulnerable detection system, we employ HGAR [10] for self-risk assessments, p-wkNN [15] to infer the probability of risk guarantee relationships. The proposed methods are utilized for the final vulnerable SME detection. During implementation, we use the Drools [22] on Apache Flink as the rule engine, in which the hot data are stored in Redis [23]. We employ neo4j as the graph database, visualize the graph by open-source software package D3.js and layout ForceAtlas2 [24]. The training model and system implementation are written in Python, Java, and Scala.

Customer Profile Date Loan Card ID Customer ID Sector Capital Registered Key: Customer ID	Loan Account Info Date Loan Card ID Customer ID Loan Contract ID Guarantee Type Key: Loan Card ID	Repayment Status Date Loan Card ID Customer ID Repayment Amount Repayment Interest Key: Loan Card ID	Guarantee Profile Guarantee Contract ID Guarantee ID Amount Key: Guarantee ID
Guarantee Contract Date Loan Contract ID Guarantee Contract ID Guarantee ID Start Date End Date Guarantee Contract ID	Loan Contract Date Customer ID Loan Contract ID Guarantee Type Start Date End Date Key: Loan Card ID	Default Status Date Loan Card ID Customer ID Default Amount Default Interest Key: Loan Card ID	Customer Credit Customer ID Rating Key: Customer ID
			Guarantee Relationship Start Time End Time Guarantee Contract ID Guarantee Contract ID

Figure 9: Overview of data association

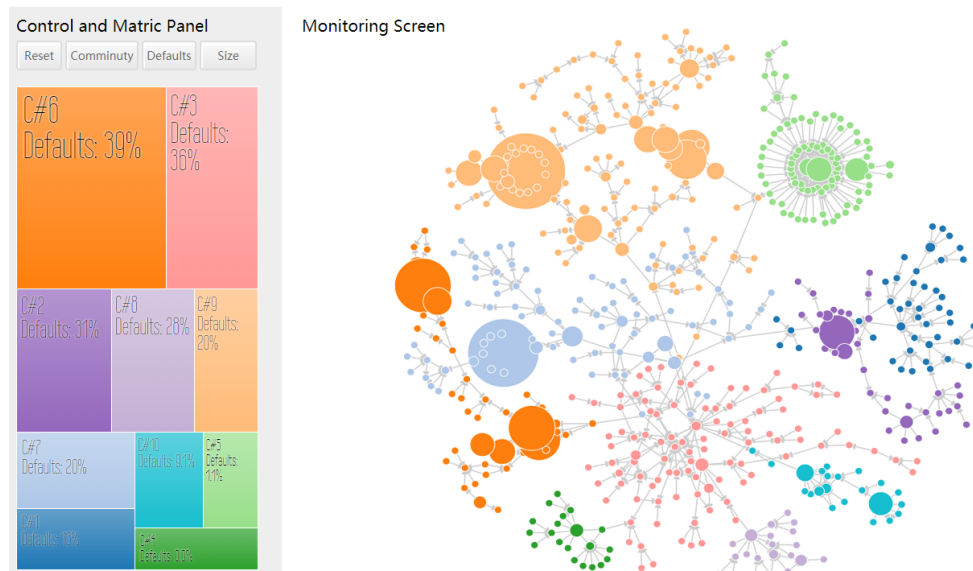


Figure 10: UI of deployed loan management system

System Deployment. Our proposed VulnDS is deployed in a loan management system of our collaborated bank. Figure 10 presents the system interface and main components, where the left part of Figure 10 presents the control and metric panel, including the risk statistics of each of the loan communities and control menus. The right part of Figure 10 displays the loan status monitoring screen. The node size indicates the delinquent probability of each company, which is dynamic and changes periodically according to the time window. Thus, risk managers could focus on risky and dominant companies.

5.2 Case Study for Loan Default Prediction

In this section, we conduct the case studies based on the deployed system. We directly observe labels from real-world behavior and validate the prediction result with the tagged labels. In the previous research, many machine learning based methods are developed for loan default prediction. To further demonstrate the performance of proposed methods, we compare the proposed methods with some baseline approaches, which are designed for the default prediction task for real-world system. The baseline methods include Wide [25], Wide and Deep [26], CNN-max [27], GBDT [28], crDNN [29], INDDP [15], HGAR [10]. We also employ betweenness [30], PageRank [31], k -core [32] and influence maximization (InfMax) [14] methods as baselines. We conduct the experiments over real-world dataset, i.e., Guarantee dataset, which spans 4 years, from 2012 to 2016. As observed, most of the loans are repaid monthly. Hence, we aggregate the behavior features within one-month time window and mark the delinquency loans as the target label for

Table 3: Results of default prediction

	AUC(2014)	AUC(2015)	AUC(2016)
Wide	0.75509	0.77751	0.78195
Wide & Deep	0.76464	0.79825	0.81053
GBDT	0.77263	0.80627	0.81182
CNN-max	0.77645	0.80049	0.81492
crDNN	0.77429	0.79565	0.81054
INDDP	0.79015	0.80927	0.81588
HGAR	0.81310	0.80988	0.81875
Betweenness	0.60649	0.60577	0.60095
PageRank	0.61359	0.61643	0.62475
K-core	0.65551	0.66281	0.66816
InfMax	0.70255	0.70927	0.71132
BSRBK	0.82367	0.82835	0.83709
BSR	0.82539**	0.83004**	0.83917**

the month. The records of 2012 are used as the training data. Then, we predict the defaults over the next three years. For the baseline methods, the training data is used to train the prediction models. For our methods, the training data is used to train the probabilities involved in the networks, which details are shown in our previous research [15].

The results are shown in Table 3, where AUC (Area Under the Curve) for each year is reported. As we can see, GBDT and Wide & Deep outperform the Wide model, because of the increase of model capacity. INDDP and HGAR are shown to be competitive across all the baselines. Betweenness and PageRank are close to each other, which do not perform satisfactory. InfMax and K-core are much better which are still suboptimal. BSR and BSRBK surpasses all the other approaches, which means the graph structure and default diffusion properties are effective for default prediction tasks. BSR is slightly better than BSRBK, because it can offer tight theoretical guarantee.

6 Related Work

Uncertain graph. The uncertain (probabilistic) graph, where each node or edge may appear with a certain probability, has been widely used to model graphs with uncertainty in a wide spectrum of graph applications. A large number of classical graph problems have been studied in the context of probabilistic graphs. For instance, [33] investigates the distance-constraint reachability problem in probabilistic graph. [4] introduces a framework to address the k nearest neighbors (kNN) queries on probabilistic graphs. [34] investigate the reliability problem based on conditional probability. In [5], authors provide a comprehensive comparison between different reliability algorithms. The problem of vulnerable nodes detection has been investigated in the context of network reliability (e.g., [35, 36, 37]). Nevertheless, their models are inherently different with ours, and the techniques cannot be trivially applied. The problem investigated in this paper is similar to the study of node influence under the independent cascade (IC) model [14, 6] in the sense that the influence of a node can be modeled by possible world semantics. Although a large body of works (e.g., [14, 38, 39, 40]) have been developed for the problem of influence maximization under the IC model, their proposed techniques cannot be applied to our problem due to the inherent difference between the two problems. Firstly, the nodes in IC model do not carry any probability. Secondly, their focus is to select k nodes such that the spread of influence is maximized. While we aim to find k nodes with largest default probabilities.

Credit evaluation. Consumer credit risk evaluation is often technically addressed in a data-driven fashion and has been extensively investigated [41, 42]. Since the seminal “Partial Credit” model [43], numerous statistical approaches have been introduced for credit scoring: logistic regression, k-NN, neural network, and support vector machine. More recently, [41] presents an in-depth analysis on interpreting the learned knowledge embedded in neural networks by using explanatory rules, and discusses how to visualize these rules. Researchers combine debt-to-income ratio with consumer banking transactions and use a linear regression model with time-windowed dataset to predict the default rates in a short future. They claim an 85% default prediction accuracy and can save costs of between 6% and 25% [44].

Diffusion in finance. The relationship between network structure and financial system risk has been carefully studied and several insights have been drawn. Network structure has little impact on system welfare, but it is important in determining systemic risk and welfare in short-term debt [45]. Network theory attracts more attention after the 2008

global financial crisis. The crisis brought about by Lehman Brothers infects connected corporations, which is similar to the 2002 Severe Acute Respiratory Syndrome (SARS) epidemic. Both of them start from small damages, but hit a networked society and cause serious events [46, 47]. The dynamic network produced by bank overnight funds loans may be an alert of the crisis [48]. Moreover, research that aims to understand individual behavior and interactions in the social network has also attracted extensive attention [49, 50]. Although preliminary efforts have been made using network theory to understand fundamental problems in financial systems [51], there is little work on system risk analysis in networked-guarantee loans [52].

7 Conclusion

In this paper, we investigate the problem of top- k vulnerable nodes detection over uncertain graphs, which is very important for risk management in real-world applications. We formally define the problem by considering both self-risk probability of the nodes and the prorogation probability of defaults among graph nodes. Due to the hardness of the problem, a sampling based method is developed with tight theoretical guarantee. To scale for large networks, effective pruning techniques and advanced sampling method are proposed with rigorous theoretical analysis. To further accelerate the search, a bottom- k based approach is presented. We conduct extensive experiments over real-world datasets to demonstrate the efficiency and effectiveness of proposed techniques. Moreover, the proposed techniques are integrated into our loan risk control system, which is deployed in real financial environment. Through case studies, we further verify the advantages of proposed model.

References

- [1] Rong-Hua Li, Qiangqiang Dai, Guoren Wang, Zhong Ming, Lu Qin, and Jeffrey Xu Yu. Improved algorithms for maximal clique search in uncertain networks. In *ICDE*, 2019.
- [2] Wenjie Zhang, Xuemin Lin, Ying Zhang, Ke Zhu, and Gaoping Zhu. Efficient probabilistic supergraph search. *TKDE*, 28(4), 2016.
- [3] Paolo Boldi, Francesco Bonchi, Aristides Gionis, and Tamir Tassa. Injecting uncertainty in graphs for identity obfuscation. *PVLDB*, 5(11), 2012.
- [4] Michalis Potamias, Francesco Bonchi, Aristides Gionis, and George Kollios. K-nearest neighbors in uncertain graphs. *PVLDB*, 3(1-2):997–1008, 2010.
- [5] Xiangyu Ke, Arijit Khan, and Leroy Lim Hong Quan. An in-depth comparison of st reliability algorithms over uncertain graphs. *PVLDB*, 12(8):864–876, 2019.
- [6] Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. Influence maximization on social graphs: A survey. *TKDE*, 30(10), 2018.
- [7] Dawei Cheng, Zhibin Niu, and Liqing Zhang. Delinquent events prediction in temporal networked-guarantee loans. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
- [8] Ming Jian and Ming Xu. Determinants of the guarantee circles: The case of chinese listed firms. *Pacific-Basin Finance Journal*, 20(1):78–100, 2012.
- [9] Dinny Mcmahon. Loan ‘guarantee chains’ in china prove flimsy. *The Wall Street Journal*, 27, 2014.
- [10] Dawei Cheng, Yi Tu, Zhen-Wei Ma, Zhibin Niu, and Liqing Zhang. Risk assessment for networked-guarantee loans using high-order graph attention representation. In *IJCAI*, pages 5822–5828, 2019.
- [11] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. In *SIGMOD*, 1987.
- [12] Zhibin Niu, Runlin Li, Junqi Wu, Dawei Cheng, and Jiawan Zhang. iconviz: Interactive visual exploration of the default contagion risk of networked-guarantee loans. In *VAST*, pages 84–94, 2020.
- [13] Dawei Cheng, Xiaoyang Wang, Ying Zhang, and Liqing Zhang. Risk guarantee prediction in networked-loans. In Christian Bessiere, editor, *IJCAI*, pages 4483–4489, 2020.
- [14] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, 2003.
- [15] Dawei Cheng, Zhibin Niu, Yi Tu, and Liqing Zhang. Prediction defaults for networked-guarantee loans. In *24th International Conference on Pattern Recognition*, pages 361–366, 2018.
- [16] Arijit Khan, Francesco Bonchi, Aristides Gionis, and Francesco Gullo. Fast reliability search in uncertain graphs. In *EDBT*, pages 535–546, 2014.

- [17] Edith Cohen and Haim Kaplan. Summarizing data using bottom-k sketches. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 225–234, 2007.
- [18] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *SODA*, pages 946–957, 2014.
- [19] Kartik Anand, Ben Craig, and Goetz Von Peter. Filling in the blanks: Network structure and interbank contagion. *Quantitative Finance*, 15(4):625–636, 2015.
- [20] Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang. Credit card fraud detection using convolutional neural networks. In *International Conference on Neural Information Processing*, pages 483–490, 2016.
- [21] Bernd Engelmann and Robert Rauhmeier. *The basel II risk parameters: estimation, validation, stress testing-with applications to loan risk management*. Springer Science & Business Media, 2011.
- [22] Ei Ei Thu and Nwe Nwe. Transforming model oriented program into android source code based on drools rule engine. *Journal of Computer and Communications*, 5(03):49, 2017.
- [23] Ming Xu, Xiaowei Xu, Jian Xu, Yizhi Ren, Haiping Zhang, and Ning Zheng. A forensic analysis method for redis database based on rdb and aof file. *Journal of Computers*, 9(11):2538–2544, 2014.
- [24] M Jacomy, S Heymann, T Venturini, and M Bastian. Forceatlas2, a figure layout algorithm for handy network visualization. *Sciences Po*, 44, 2012.
- [25] HB McMahan. Follow-the-regular ized-leader and mil-ror descent: Equivalence theorems and l1 regularization. *Journal of Machine Learning Research Proceedings Trade*, 15:525–533, 2011.
- [26] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhya, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10, 2016.
- [27] Lei Zheng, Vahid Noroozi, and Philip S Yu. Joint deep modeling of users and items using reviews for recommendation. In *ACM International Conference on Web Search and Data Mining*, 2017.
- [28] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.
- [29] Fei Tan, Xiurui Hou, Jie Zhang, Zhi Wei, and Zhenyu Yan. A deep learning approach to competing risks representation in peer-to-peer lending. *IEEE transactions on neural networks and learning systems*, 2018.
- [30] Sara Mumtaz and Xiaoyang Wang. Identifying top-k influential nodes in networks. In *CIKM*, 2017.
- [31] Amy N Langville and Carl D Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004.
- [32] Chen Chen, Qiuyu Zhu, Renjie Sun, Xiaoyang Wang, and Yanping Wu. Edge manipulation approaches for k-core minimization: Metrics and analytics. *TKDE*, 2021.
- [33] Ruoming Jin, Lin Liu, Bolin Ding, and Haixun Wang. Distance-constraint reachability computation in uncertain graphs. *Proceedings of the VLDB Endowment*, 4(9):551–562, 2011.
- [34] Arijit Khan, Francesco Bonchi, Francesco Gullo, and Andreas Nufer. Conditional reliability in uncertain graphs. *TKDE*, 30(11), 2018.
- [35] Shudong Li, Lixiang Li, Xinran Liu, and Yixian Yang. Identifying vulnerable nodes of complex networks in cascading failures induced by node-based attacks. *Mathematical Problems in Engineering*, 2013:938398, 09 2013.
- [36] Arunabha Sen, Anisha Mazumder, Joydeep Banerjee, Arun Das, and Randy Compton. Identification of K most vulnerable nodes in multi-layered network using a new model of interdependency. In *IEEE INFOCOM Workshops*, pages 831–836, 2014.
- [37] Hale Cetinay, Karel Devriendt, and Piet Van Mieghem. Nodal vulnerability to targeted attacks in power grids. *Applied Network Science*, 3(1):34, 2018.
- [38] Xiaoyang Wang, Ying Zhang, Wenjie Zhang, Xuemin Lin, and Chen Chen. Bring order into the samples: A novel scalable method for influence maximization. *TKDE*, 29(2):243–256, 2017.
- [39] Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*, 2015.
- [40] Xiaoyang Wang, Ying Zhang, Wenjie Zhang, and Xuemin Lin. Efficient distance-aware influence maximization in geo-social networks. *TKDE*, 29(3):599–612, 2016.
- [41] Bart Baesens, Rudy Setiono, Christophe Mues, and Jan Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management science*, 49(3):312–329, 2003.

- [42] David J Hand and William E Henley. Statistical classification methods in consumer credit scoring: a review. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 160(3):523–541, 1997.
- [43] Geoff N Masters. A rasch model for partial credit scoring. *Psychometrika*, 47(2):149–174, 1982.
- [44] Amir E Khandani, Adlar J Kim, and Andrew W Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010.
- [45] Franklin Allen, Ana Babus, and Elena Carletti. Financial connections and systemic risk. Technical report, National Bureau of Economic Research, 2010.
- [46] Spiros Bougheas and Alan Kirman. Complex financial networks and systemic risk: A review. In *Complexity and Geographical Economics*, pages 115–139. 2015.
- [47] Dawei Cheng, Zhibin Niu, and Yiyi Zhang. Contagious chain risk rating for networked-guarantee loans. In *KDD*, pages 2715–2723, 2020.
- [48] Véronique Van Vlasselaer, Jan Meskens, Dries Van Dromme, and Bart Baesens. Using social network knowledge for detecting spider constructions in social security fraud. In *ASONAM*, pages 813–820, 2013.
- [49] Jukka-Pekka Onnela et al. *Complex networks in the study of financial and social systems*. Helsinki University of Technology, 2006.
- [50] Jiezhong Qiu, Yixuan Li, Jie Tang, Zheng Lu, Hao Ye, Bo Chen, Qiang Yang, and John E Hopcroft. The lifecycle and cascade of wechat social messaging groups. In *WWW*, pages 311–320, 2016.
- [51] Wing S Chow and Lai Sheung Chan. Social network, social trust and shared goals in organizational knowledge sharing. *Information & Management*, 45(7):458–465, 2008.
- [52] Xiangfeng Meng, Yunhai Tong, Xinhai Liu, Yiren Chen, and Shaohua Tan. Netrating: Credit risk evaluation for loan guarantee chain in china. In *Pacific-Asia Workshop on Intelligence and Security Informatics*, pages 99–108, 2017.