Information Sciences 278 (2014) 685–702

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Measuring the impact of MVC attack in large complex networks

Rong-Hua Li [a,b,*], Jeffrey Xu Yu [c], Xin Huang [c], Hong Cheng [c], Zechao Shang [c]

[a] Guangdong Province Key Laboratory of Popular High Performance Computers, Shenzhen University, China
[b] East China Normal University, China
[c] The Chinese University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 17 April 2013
Received in revised form 9 March 2014
Accepted 13 March 2014
Available online 29 March 2014

Keywords:
Complex network
MVC attack
FM sketch
(Adaptive) submodularity

ABSTRACT

Measuring the impact of network attack is an important issue in network science. In this paper, we study the impact of maximal vertex coverage (MVC) attack in large complex networks, where the attacker aims at deleting as many edges of the network as possible by attacking a small fraction of nodes. First, we present two metrics to measure the impact of MVC attack. To compute these metrics, we propose an efficient randomized greedy algorithm with near-optimal performance guarantee. Second, we generalize the MVC attack into an uncertain setting, in which a node is deleted by the attacker with a prior probability. We refer to the MVC attack under such uncertain environment as the probabilistic MVC attack. Based on the probabilistic MVC attack, we propose two adaptive metrics, and then present an adaptive greedy algorithm for calculating such metrics accurately and efficiently. Finally, we conduct extensive experiments on 20 real datasets. The results show that P2P and co-authorship networks are extremely robust under the MVC attack while both the online social networks and the Email communication networks exhibit vulnerability under the MVC attack. In addition, the results demonstrate the efficiency and effectiveness of the proposed algorithms for computing the proposed metrics.

1. Introduction

Networks are ubiquitous. Many practical systems in nature and society can be characterized by the network. Examples include (online) social networks, computer networks, Internet, biological networks, transportation networks, and so on. After the seminal work by Watts and Strogatz [51] and Barabási and Albert [2], complex networks have attracted increasing attention in both industry and research communities in the last decade. The studies of complex network mainly focus on investigating the underlying organizing principles, the function, and the dynamics of the network.

However, networks are very often attacked by malicious users. An important issue in network science is to study the robustness of a network subject to nodes or links errors [1,52]. Many real-world examples are briefly described below. In airline network, an important question is how the operational ability of the network is affected given that a certain airports are closed. In computer networks, a key problem is to study how the communication capacity changes when some computers in the network are attacked by the hackers. In P2P networks, a crucial issue is to study how the communication ability of the network is affected when a certain number of peers depart from the network.

In practice, a network attacker typically has a small budget of k nodes (edges) to attack due to resource limitation. The attacker aims at maximizing some utility functions after attacking k nodes (edges). Many previous studies have been focused

* Corresponding author. Address: East China Normal University, China. Tel.: +86 13510721660.
E-mail address: lironghuascut@gmail.com (R.-H. Li).

http://dx.doi.org/10.1016/j.ins.2014.03.085
0020-0255/

on such nodes (edges) attack problems. For example, Albert et al. [1] studied the robustness of a network by considering the diameter of the network after deleting a small fraction of nodes. In [1], the utility function of the attacker corresponds to the diameter of the network, and the attacker aims to maximize the diameter of the network, which will make the network as less cohesive as possible. Many subsequent studies [9,8,10] followed this framework to study how a network is affected by nodes or edges errors. However, most of them focus on deriving analytical solutions on the basis of some specific random graph models. More recently, Schneider et al. [45] investigated how the size of the giant connected component changes subject to nodes deletions. Clearly, in [45], the utility function of the attacker is the inverse of the size of the maximal connected component of the network, and the attacker aims to minimize the size of the maximal connected component of the network.

Instead of the diameter and the size of maximal connected component, in this paper, we consider another important metric, i.e., the number of residual edges of the network after attacking. Specifically, in our model, the attacker's utility function is equal to the number of edges that are removed. The attacker aims at maximizing the number of edges that are deleted after attacking a small fixed budget of k nodes of the network. Or, equivalently, the attacker wants to minimize the number of residual edges of a network after attacking. Clearly, this problem is equivalent to the maximal vertex coverage (MVC) problem on networks [21]. We thus refer to such type of attack as the MVC attack. Note that the number of residual edges is a very natural and intuitive metric to measure the function and performance of the network. Intuitively, after the removal of k nodes, the network with a large number of residual edges implies that the function and performance of the network are not extensively damaged. There are many practical applications that can suffer from such MVC attack. For instance, in computer networks, the hacker may want to attack k workstations so as to minimize the number of surviving links in the network. In online social networks, the attacker may want to target k users by providing some incentives to persuade them to leave the social network so as to minimize the number of residual social ties of the network. Such incentive-based attacks indeed encounter in real-world online social networks. For example, a recent news in CNET (http://news.cnet.com/delete-10-facebook-friends-get-a-free-whopper/) reports an attack event. The fast-food company Burger King developed a Facebook application, namely Whopper Sacrifice, giving a Facebook user a free coupon for a free hamburger if he/she deletes 10 people from his/her friends list. By statistics, 82,771 Facebook users participated in this campaign, and 233,906 Facebook friendships were deleted. After then, Facebook closed this application due to a large number of link deletions. In addition, on the positive side, the MVC attack could be used to fight terrorism. For example, given a terrorist network, we may want to attack a small number of terrorists so that the number of residual ties in the network is minimal. As a consequence, it is important to investigate the impact of MVC attack in a network.

To study impact of the MVC attack, we present two new metrics of the network, named *k-MVC-impact* and *cumulative k-MVC-impact*. More specifically, *k-MVC-impact* is defined by the minimal fraction of the residual edges after removing k nodes, and *cumulative k-MVC-impact* is the average *k-MVC-impact* from $k = 1$ to k. To compute the two proposed metrics, we propose an optimal dynamic programming algorithm for tree-like networks. For general networks, by using the submodularity of the MVC problem, we present a randomized greedy algorithm with near-optimal approximation guarantee for calculating our metrics efficiently. To the best of our knowledge, this work is the first work to measure the impact of MVC attack in complex networks. Besides the MVC attack, we also introduce a probabilistic MVC problem where a node is not always successfully removed by the attacker. Instead, we consider that the node is removed by the attacker with a prior probability. In terms of probabilistic MVC attack, we also propose two new metrics, called *adaptive k-MVC-impact* and *adaptive cumulative k-MVC-impact*. We show that the probabilistic MVC problem can be formulated as an adaptive submodular function maximization problem [19]. Based on this, we present a near-optimal adaptive greedy algorithm to calculate *adaptive k-MVC-impact* and *adaptive cumulative k-MVC-impact* efficiently. Finally, we conduct extensive experimental studies on 20 real-world datasets. The results show that the P2P and co-authorship networks are extremely robust subject to MVC attack, whereas the online social networks, the Email communication networks, as well as the web graph are shown to be very vulnerable subject to MVC attack. Also, the results confirm the effectiveness and efficiency of the proposed algorithms for computing the proposed metrics.

The remainder of this paper is organized as follows. We describe the problem and propose two metrics for studying the impact of MVC attack in Section 2. We then present two new algorithms for computing the proposed metrics in Section 3. In Section 4, we propose two new metrics w.r.t. the probabilistic MVC attack and develop an efficient adaptive greedy algorithm for calculating them accurately. We conduct extensive experimental studies in Section 5. Finally, we survey the related work in Section 6 and conclude this work in Section 7.

2. Problem formulation

Consider an undirected network $G = (V, E)$, where V denotes a set of nodes and E denotes a set of undirected edges between the nodes. Let $n = |V|$ and $m = |E|$ be the number of nodes and the number of edges in G, respectively. The problem that we address in this paper is to measure the impact of MVC attack in a network.

Specifically, we consider the following setting. Assume that there is an attacker who wants to attack a network, and the attacker has a budget of k nodes to attack. If a node is attacked by the attacker, then the node and its incident edges will be removed from the network. The attacker aims to maximize some utility functions by attacking k nodes. In this paper, we introduce a utility function for the attacker. That is, the number of edges that are removed by attacking k nodes. In other words, the goal of the attacker is to maximize the number of edges that are removed after deleting k nodes. Note that this

problem is equivalent to the maximal vertex coverage (MVC) problem [21] which aims to select k nodes that cover as many edges as possible. Therefore, we refer to such an attack as the MVC attack. Formally, let S ($S \subseteq V$) be a set of nodes, and $F(S)$ be the number of edges that are removed after deleting the nodes in S. Then, the MVC attack problem can be formulated as

$$\begin{aligned} &\max_{S\subseteq V} F(S) \\ &\text{s.t.} \quad |S| \leqslant k. \end{aligned} \tag{1}$$

Let $F^*(S)$ be the optimal solution for Eq. (1). Then, we define a metric called *k-MVC-impact* of a network G as follows.

Definition 2.1. Given a network $G = (V, E)$, the *k-MVC-impact* of G is $\sigma_k = 1 - F^*(S)/m$.

By Definition 2.1, the *k-MVC-impact* (σ_k) denotes the fraction of residual edges after removing k nodes. Intuitively, after removing k nodes, the larger the fraction of residual edges is, the more robust the network is. In addition, it can be seen that σ_k falls into the interval $[0,1]$. If $\sigma_k = 0$, we say a network is completely collapsed. We refer to the minimal k that causes $\sigma_k = 0$ as the collapsed point denoted by $\tilde{k}$. Clearly, $\tilde{k}$ equals to the minimum vertex cover number of a network. Note that σ_k measures the *point-wise* impact of MVC attack in a network. Naturally, we define the cumulative impact called *cumulative k-MVC-impact* as follows.

Definition 2.2. Given a network $G = (V, E)$, the *cumulative k-MVC-impact* of G is $\bar{\sigma}_k = \left(\sum_{i=1}^{k} \sigma_i\right)/k$.

Unlike the *k-MVC-impact*, the *cumulative k-MVC-impact* evaluates the average point-wise *k-MVC-impact*. According to Definitions 2.1 and 2.2, large σ_k and $\bar{\sigma}_k$ indicate a high robustness of the network subject to MVC attack. Note that the key subroutine to compute *cumulative k-MVC-impact* ($\bar{\sigma}_k$) is to calculate *k-MVC-impact* (σ_k). In the following section, we focus on how to compute σ_k efficiently. It is worth mentioning that, although we mainly focus on the undirected networks, the proposed metrics and techniques can be easily generalized to directed networks. This is because in our model, once a node is deleted, its incident edges are also removed. Therefore, under our model, we can ignore the directions of edges for the directed networks, and then apply the proposed metrics and techniques to measure the impact of MVC attack.

3. Algorithms

Given a network G, the key issue to evaluate *k-MVC-impact* of G is to solve the MVC attack problem (Eq. (1)). Unfortunately, finding the MVC on general networks has been known to be NP-complete [11]. Hence, there is no hope to exactly compute σ_k in polynomial time. In this section, we first propose an optimal dynamic programming algorithm for computing σ_k on trees. Then, we introduce a randomized greedy algorithm for calculating σ_k on general networks.

3.1. Algorithm for trees

Here we show that σ_k can be exactly calculated on trees in polynomial time by a dynamic programming algorithm. Consider a tree T whose root node has c children. The optimal way for finding k nodes that maximize $F(S)$ ($|S| = k$) must follow either of the following two cases. The first case is that we select the root node into the set S and then recurse on the children with a budget of $k-1$. The second case is that we do not choose the root node and instead recurse on the children with a budget of k. However, such recursion will result in very high computational cost, because the recursion needs to partition the c children into k or $k-1$ parts in all possible ways.

To reduce the computational cost, we transform the tree T into a binary tree $\mathcal{T}$ in the following way. For every node v in T, assume the children of v are $v_1, \ldots, v_c$. If $c \leqslant 2$, we do nothing. Otherwise, we replace v by a binary tree with leaves $v_1, \ldots, v_c$. Specifically, let v_1 be the left child of v. Create a virtual node w_1 and let it be the right child of v. Then, let the rest children of v be the children of w_1. Repeat this procedure until every node has at most two children. Notice that the number of nodes in $\mathcal{T}$ is at most twice the number of nodes in T and the depth of $\mathcal{T}$ is at most a factor of $\log_2 d_{\max}$ larger than the depth of T. Here $d_{\max}$ denotes the maximal out-degree of a node in T. Similar constructions have been applied for different applications [28,29].

Based on the above transformation, we can design a dynamic programming (DP) algorithm on $\mathcal{T}$. Let $J(v, S, k)$ be the optimal solution in the subtree rooted by v with a budget k, $L(v)$ and $R(v)$ be the left and right child of node v respectively. Then, the recursive equation of the DP algorithm is given by

$$J(v,S,k) = \max\left\{ \max_{i=0,1,\ldots,k} \{J(L(v),S,i) + J(R(v),S,k-i)\}, C(v,S) + \max_{i=0,1,\ldots,k-1} \{J(L(v),S\cup\{v\},i) + J(R(v),S\cup\{v\},k-1-i)\} \right\}, \tag{2}$$

where $C(v, S)$ is a cost function and it is defined as follows. For each node v in $T \cup \mathcal{T}$, we set $C(v,S) = F(S \cap \{v\}) - F(S)$, where S denotes the current optimal solution and $F(S)$ is evaluated on the original tree T. In other words, $C(v, S)$ denotes the gain of the objective function after selecting node v into the solution set S. For each virtual node v in $\mathcal{T}$, we set $C(v,S) = -\infty$ to ensure that the virtual nodes will not be selected into the optimal solution S. The first term in Eq. (2) represents the case that the node v is not selected into the solution S. The second term denotes the case that the node v is added into the solution

S. The optimal solution can be obtained by invoking $J(v,S,k)$, where v is the root node of the tree $\mathcal{T}$ and $S=\emptyset$. One can easily show that the optimal solution obtained by the DP algorithm on $\mathcal{T}$ is the optimal solution of the MVC attack problem (Eq. (1)) on T.

We analyze the time complexity of the DP algorithm as follows. First, the time complexity for building the binary tree $\mathcal{T}$ is $O(n\log_2 d_{\max})$. Second, for each node in the tree $\mathcal{T}$, we need to evaluate the recursion $O(k)$ times, and each evaluation takes at most $O(n)$ time. There are $O(n\log_2 d_{\max})$ number of nodes in $\mathcal{T}$. Consequently, the time complexity of the DP algorithm is $O(kn^2\log_2 d_{\max})$. For the space complexity, the algorithm needs to store the trees T, $\mathcal{T}$, and maintains all the $J(v,S,k)$, which results in $O(kn^2\log_2 d_{\max})$ space complexity.

Remark. The proposed tree transformation method is based on a paradigm of "trading space for time". Note that in the transformed binary tree, we include a small number of virtual nodes to speedup the dynamic programming algorithm. Moreover, we show that such a number can be bounded, and the additional space overhead is only twice than the space complexity of the original tree. In addition, we emphasize that our purposes of studying impact of MVC attack on trees are twofold. On the one hand, tree is a powerful model to characterize the hierarchy structure in many real-world applications. Such applications include the evolutionary tree in biology, social hierarchy structure in social networks, as well as XML data. The proposed study could be useful for these applications where it needs to measure the impact of MVC attack. On the other hand, for the trees, we have a nice polynomial-time algorithm to calculate the proposed metrics. However, for general networks, computing the proposed metrics is NP-hard, and thus we has to resort to approximation algorithms. Therefore, the algorithm for trees can serve as a very useful baseline to evaluate the practical performance of the approximation algorithms when they are performed on trees.

3.2. Algorithm for general networks

In this subsection, we present a randomized greedy algorithm with near-optimal performance guarantee for computing σ_k on general networks efficiently. First, we briefly describe the concept of nondecreasing submodular set function. Let A be a finite set. A set function F defined on the subsets of A is a nondecreasing submodular function if the following condition holds. For any subsets B and C such that $B\subseteq C\subseteq A$, and for any element $j\notin C$, we have $\rho_j(B)\geqslant\rho_j(C)\geqslant 0$, where $\rho_j(B)$ represents the marginal gain and it is defined as $\rho_j(B)=F(B\cup\{j\})-F(B)$.

It is easy to check that $F(S)$ is a nondecreasing monotone submodular set function. Based on this, there exists a greedy algorithm for solving the MVC problem (Eq. (1)) efficiently. In particular, the greedy algorithm works in k rounds. In each round, the algorithm finds the node with the maximal marginal gain ($\rho_j(S)$) and adds it to the optimal node set S, where S is initialized to be an empty set. By a celebrated result [42], this greedy algorithm can achieve a $1-1/e$ approximate ratio. The time complexity of the greedy algorithm is $O(km)$, because the algorithm needs to visit all the edges to find the node with the maximal marginal gain in the worst case. Below, we will propose a more efficient randomized greedy algorithm using the well-known Flajolet-Martin (FM) sketch [17].

The FM sketch is a probabilistic counting data structure and it can be utilized to estimate the cardinality of a multi-set [17]. Let N be the cardinality of a multi-set A. Then, the FM sketch only uses $\log N+c$ bits for estimating N accurately, where c is a small constant. In particular, the FM sketch is a bitmap with size $l=\log N+c$. There exists a hash function $h:A\to\{1,\ldots,l\}$, mapping an element a ($a\in A$) to a bit i ($i\in\{1,\ldots,l\}$) in the bitmap with probability $\Pr(h(a)=i)=1/(2^{i+1})$. At the beginning, all the bits in the bitmap are set to 0. Then, for processing an element a ($a\in A$), we set the corresponding $h(a)$-th bit of the bitmap to 1. Finally, an asymptotically unbiased estimator for the cardinality N can be obtained by $2^z/0.77351$, where z denotes the position of the least-significant zero bit in the bitmap. Another important property of the FM sketch is that it can be easily used to estimate the cardinality of the union of two multi-sets if these two multi-sets come from the same domain. Specifically, we construct two FM sketches with the same size for two multi-sets respectively. To estimate the cardinality of the union of two multi-sets, we only need to do a bitwise-OR between the two FM sketches, and then estimate the cardinality based on the resulting FM sketch. To enhance the estimation accuracy, we can make use of multiple hash functions. For convenience, we only consider one hash function to describe our algorithm. In addition, it is worth mentioning that there also exist many other probabilistic counting structures, such as Loglog sketch [13] and Hyper Loglog sketch [16]. Here we select to use FM sketch because it is easy to be implemented, and both its efficiency and its estimating accuracy are desirable as shown in our experiments.

The key idea of our algorithm is described as follows. For each node u, we create an FM sketch to sketch the incident edges of u and use it to estimate $F(\{u\})$. Then, for any set S, $F(S)$ can be calculated by

$$F(S)=\left|\bigcup_{u\in S}E(\{u\})\right|, \tag{3}$$

where $E(\{u\})$ denotes the set of incident edges of node u. Note that $E(\{u\})$ can be represented by an FM sketch. As a result, for any set S, we can estimate $F(S)$ by performing $|S|$ times bitwise-OR operation. Our algorithm is described in Algorithm 1. Firstly, Algorithm 1 creates an FM sketch for each node $v_i\in V$ (lines 2–5). In particular, for each node v_i, we initialize a

bitmap FM[i], i.e., set all the bits of FM[i] to 0 (line 3). For all the incident edges of node v_i, we insert them into the bitmap FM[i] by setting the corresponding bits to 1 (lines 4–5). Secondly, Algorithm 1 greedily chooses k nodes based on their approximate marginal gain (lines 6–22). Specifically, we create two FM sketches CFM and OFM and use them to estimate the current optimal solution and the current marginal gain, respectively. Algorithm 1 works in k rounds. In each round, it selects the node with the maximal approximate marginal gain (lines 12–19). To compute the approximate marginal gain of node v_i (denoted by $\hat{\rho}_i$), we only need to do a bitwise-OR between the FM sketches CFM and FM[i] (line 13), which results in the FM sketch OFM. Then, we can use the standard unbiased estimator to estimate $\hat{\rho}_i$ for node v_i (lines 14–15). After finding the node with the maximal approximate marginal gain, we need to update the answer set S and the FM sketch CFM. Note that we only need to do a bitwise-OR between the FM sketches CFM and FM[Idx] to update the CFM (lines 19–22). Here FM[Idx] denotes the FM sketch of the node v_{Idx} which achieves the maximal approximate marginal gain. Finally, Algorithm 1 outputs the answer set S and the approximate σ_k (line 23). Notice that to calculate *cumulative k-MVC-impact* $\bar{\sigma}_k$ we do not need to invoke Algorithm 1 k times, but invoke Algorithm 1 with parameter k only once. Because we can record all the F (line 22) obtained in each round and compute *cumulative k-MVC-impact*. Additionally, we can use the so-called CELF framework [32] to accelerate both the original greedy algorithm and our randomized greedy algorithm.

Algorithm 1. The Randomized Greedy Algorithm

Input: Network $G = (V, E)$ and k.
Output: A set S with k nodes, σ_k
1: Let $h : \{e_1, \ldots, e_m\} \rightarrow \{1, \ldots, l\}$ be the hash function that maps the edges to a position of the BITMAP, here $l = \log m + c$ denotes the size of the BITMAP;
2: **for** each node $v_i \in V$ **do**
3: Initialize a BITMAP FM[i] $\leftarrow$ 0;
4: each incident edge e of v_i **do**
5: Set the $h(e)$-bit of FM[i] to 1;
6: **end for**
7: **end for**
8: $S \leftarrow \emptyset$;
9: Create two FM sketches CFM $\leftarrow$ 0, OFM $\leftarrow$ 0;
10: $F \leftarrow 0$; **do**
11: **for** iter = 1 to k
12: max $\leftarrow -1$;
13: $Idx \leftarrow 0$;
14: **for** each node $v_i \in (V \setminus S)$ **do**
15: OFM $\leftarrow$ (CFM) bitwise-OR (FM[i]);
16: Let z be the position of the least-significant 0 bit in OFM;
17: $\hat{\rho}_i \leftarrow 2^z/0.77351 - F$;
18: **if** $\hat{\rho}_i >$ max **then**
19: max $\leftarrow \hat{\rho}_i$;
20: $Idx \leftarrow i$;
21: **end if**
22: **end for**
23: $S \leftarrow S \cup \{v_{Idx}\}$;
24: CFM $\leftarrow$ (CFM) bitwise-OR (FM[Idx]);
25: Let z be the position of the least-significant 0 bit in CFM;
26: $F \leftarrow 2^z/0.77351$;
27: **end for**
28: **return** S and $1 - F/m$;

Theoretically, by a similar analysis as in [20], Algorithm 1 can achieve a $1 - 1/e - \epsilon$ approximate ratio with high probability for computing the σ_k on general networks. The reason is because the FM sketch estimates the marginal gain $\rho_i(S)$ of any set S within an ϵ error bound with high probability [17]. The time complexity of Algorithm 1 is $O(kn + m)$. First, Algorithm 1 takes $O(m)$ time to initialize the FM sketches for every node (lines 2–5). Second, Algorithm 1 uses $O(kn)$ time to compute the σ_k. The rationale is that the bitwise-OR (line 13) and the estimation step (lines 14–15) can be done in constant time [43]. We emphasize that $O(kn + m)$ is more efficient than $O(km)$ when k cannot be ignored. Another advantage of Algorithm 1 is that the FM sketches for every node can be built offline. Assume that we have built the FM sketches for every node of a given network G. Then, for any given k, Algorithm 1 can compute the corresponding σ_k in $O(kn)$ time. However, the original greedy algorithm still needs $O(km)$ time complexity for computing σ_k. For the space complexity, Algorithm 1 needs

to store the network G which takes $O(m+n)$ space complexity. In addition, Algorithm 1 maintains $O(n)$ FM sketches which take $O(n\log m)$ bits, because each FM sketch only takes $O(\log m)$ bits. The size of $O(n\log m)$ bits can be dominated by the $O(m+n)$ graph size. So putting it all together, the space complexity of Algorithm 1 is $O(n+m)$, which is the same as the original greedy algorithm.

3.3. Discussions

Here we show that the proposed technique can also be generalized to the cost-based attack model. We briefly describe the cost-based attack model as follows. In the cost-based attack model, each node in the network is associated with a non-negative weight which models the cost needed to successfully remove the node. The attacker has a budget of k units cost, and its goal is to maximize the number of removed edges after attacking some nodes subject to the budget constraint. More formally, we have

$$\begin{aligned} &\max_{S\subseteq V} F(S)\\ &\text{s.t.} \quad \sum_{v\in S} c_v \leqslant k, \end{aligned} \tag{4}$$

where c_v denotes the cost of node v. Based on the optimal solution of Eq. (4), the *k-MVC-impact* and *cumulative k-MVC-impact* under the cost-based attack model can be defined in a similar way to Definitions 2.1 and 2.2. Below, we discuss how to calculate the optimal solution of Eq. (4).

Since $F(S)$ is a submodular function, the problem described in Eq. (4) is a submodular function maximization with knapsack constraint problem, which is known to be NP-hard [39]. For such a problem, there is an efficient greedy algorithm with a $(1-1/\sqrt{e})$ approximation factor [38]. Specifically, in each greedy step, without violating the budget constraint, the algorithm adds the node with the maximal ratio of the marginal gain to the cost of that node. More formally, in each greedy step, the node selection of the greedy algorithm is according to the following rule

$$\arg \max_{v\in V\setminus S, c_v+\sum_{u\in S} c_u\leqslant k} (F(S\cup\{v\})-F(S))/c_v, \tag{5}$$

where S denotes the set of selected nodes up to current greedy step. The greedy node-selection process terminates if there is no node that can be added. Let S_1 and $F(S_1)$ be the optimal solution and optimal value obtained by the greedy node-selection process, respectively. Then, the algorithm computes $F(v^*)$ according to the following equation

$$F(v^*) = \max_{v\in V, c_v\leqslant k} F(\{v\}). \tag{6}$$

Finally, the optimal solution of the greedy algorithm is obtained by

$$\arg \max_{S\in\{\{v^*\},S_1\}} F(S). \tag{7}$$

Based on such a greedy algorithm, our randomized greedy algorithm (Algorithm 1) can be easily generalized to solve the optimization problem given in Eq. (4). To this end, we need to modify the termination condition of the main loop in Algorithm 1 (line 9) to a budget constraint, and also we need to modify the node selection rule (lines 16–18) to the rule described in Eq. (5). The detailed algorithm is very similar to Algorithm 1, thus we omit it for brevity.

4. Probabilistic MVC attack

In the previous sections, we consider the problem for measuring the impact of a network under a deterministic MVC attack model. That is, if an attacker attacks a node, then the node as well as its incident edges will be removed from the network. However, in practice, the nodes in a network may be tolerant of attacks to some extent. In other words, the attacker may not always delete a node successfully. In this section, we study the problem for measuring the impact of a network under such a scenario.

Now we consider the following setting. Suppose that an attacker has a budget of k nodes to attack. If the attacker attacks a node v, then the node v will be successfully attacked by the attacker with a probability $p(v)$. If the node v is successfully attacked by the attacker, then v and its incident edges will be removed from the network. If the node v survives after attack, then we do nothing. The attacker sequentially attacks the nodes one by one and he/she can observe the previous outcomes, i.e., whether the attacked nodes are successfully removed or not. In addition, we assume that either the attacker succeeds or fails, the attacker cannot make any attempt to attack the same node again. The reason is twofold. First, if the attacker succeeds, then the node will be removed from the network, thereby there is no chance to attack it again. Second, if the attacker fails, then the attacker deems that this node is quite robust and it is very hard to be attacked, thus the attacker gives up attacking the same node again. The attacker wants to find an optimal policy for selecting nodes to attack such that he/she can maximize the expected utility function, i.e., the number of deleted edges. Clearly, such a problem requires the attacker to make decision under uncertainty. For convenience, we refer to such a problem as the probabilistic MVC problem,

as it is a probabilistic generalization of the original MVC problem. We formulate the probabilistic MVC problem as a stochastic optimization problem as follows.

Under the above attack model, each node in a network has two states 0 and 1 which signify the node is alive or deleted after attack, respectively. Let $X_i \in \{0,1\}$ be the state of node v_i and $p(v_i)$ be the probability of the deletion of node v_i. Further, we let $\phi = (X_1, X_2, \ldots, X_n)$ be a possible state of the network G and Ω be the set of all the possible states of G, i.e., $\phi \in \Omega$. In addition, we assume that the variables X_i $(i = 1, 2, \ldots, n)$ are independent. This is because the events of the nodes deleted successfully or not are typically independent of each other. Based on this, we have

$$\Pr[\phi] = \prod_{v_i \in V} p(v_i)^{X_i}(1 - p(v_i))^{1-X_i}. \tag{8}$$

Let π be a policy for choosing nodes to attack. The utility function $F : 2^n \times \Omega \to \mathbb{R}_{\geqslant 0}$ represents the number of edges that are removed after attack, which depends on which nodes we select and which state of each node is. Let $S(\pi, \phi)$ be the set of nodes selected by π under the possible state ϕ. Then, the expected utility function of a policy π is given by

$$\overline{F}(S(\pi, \phi)) = \mathbb{E}[F(S(\pi, \phi), \phi)], \tag{9}$$

where the expectation is taken w.r.t. $p(\phi)$. Based on these notations, the probabilistic MVC problem aims to find the optimal policy so as to maximize $\overline{F}(S(\pi, \phi))$. Formally,

$$\begin{aligned} &\arg\max_{\pi} \ \overline{F}(S(\pi, \phi)) \\ &\text{s.t.} \qquad |S(\pi, \phi)| \leqslant k, \end{aligned} \tag{10}$$

where $|S(\pi, \phi)|$ denotes the cardinality of the set $S(\pi, \phi)$.

Let π^* and $\overline{F}(S(\pi^*, \phi))$ be the optimal policy and the optimal value for Eq. (10), respectively. Then, similar to Definitions 2.1 and 2.2 we can define the *adaptive k-MVC-impact* and *adaptive cumulative k-MVC-impact* of a network G as follows.

Definition 4.1. Given a network $G = (V, E)$, and the probability of each node v, i.e., $p(v)$, the *adaptive k-MVC-impact* of G is $\delta_k = 1 - \overline{F}(S(\pi^*, \phi))/m$.

Definition 4.2. Given a network $G = (V, E)$, and the probability of each node v, i.e., $p(v)$, the *adaptive cumulative k-MVC-impact* of G is $\bar{\delta}_k = \left(\sum_{i=1}^{k} \delta_i\right)/k$.

Likewise, the larger δ_k and $\bar{\delta}_k$ imply the larger *adaptive k-MVC-impact* of a network. Below, we focus on developing algorithm for solving Eq. (10) by using the property of adaptive submodularity [19].

4.1. The adaptive greedy algorithm

In this subsection, we propose a near-optimal greedy algorithm for solving Eq. (10) efficiently. Below, we prove that the utility function F is an adaptive submodular function [19].

Adaptive submodularity: First, we introduce some important notations and definitions on adaptive submodularity introduced in [19]. Let ψ be a partial observation and $dom(\psi)$ be the set of nodes that have been observed. In other words, ψ captures both the nodes in $dom(\psi)$ and their corresponding states. A partial observation ψ is consistent with a possible state ϕ if and only if all the states of the nodes in $dom(\psi)$ are consistent with the states of the nodes represented by ϕ, and we denote it by $\psi \sim \phi$. Assume there is another partial observation ψ' and $dom(\psi) \subseteq dom(\psi')$. Then, ψ and ψ' are consistent if and only if all the states of the nodes in ψ are consistent with the states of the same nodes in ψ', and we denote it by $\psi \sim \psi'$. Next, we give the definition of conditional expected marginal gain as follows.

Definition 4.3 (*Conditional expected marginal gain*). Given a partial observation ψ and a node v, the conditional expected marginal gain of choosing a node v is given by

$$\Delta(v|\psi) = \mathbb{E}[F(dom(\psi) \cup \{v\}, \phi) - F(dom(\psi), \phi) | \psi \sim \phi], \tag{11}$$

where the expectation is taken w.r.t. $p(\phi|\psi)$.

Based on the conditional expected marginal gain, we give the definition of strong adaptive monotonicity and adaptive submodularity as follows.

Definition 4.4 (*Strong adaptive monotonicity*). A function $F : 2^n \times \Omega \to \mathbb{R}_{\geqslant 0}$ is strong adaptive monotone w.r.t. distribution $p(\phi)$ if the following condition holds. For all ψ, all $v \notin dom(\psi)$, and all possible state o of node v such that $\Pr[\phi(v) = o | \psi \sim \phi] > 0$, we have

$$\mathbb{E}[F(dom(\psi), \phi) | \psi \sim \phi] \leqslant \mathbb{E}[F(dom(\psi) \cup \{v\}, \phi) | \psi \sim \phi, \phi(v) = o]. \tag{12}$$

Definition 4.5 (*Adaptive submodularity*). A function $F: 2^n \times \Omega \to \mathbb{R}_{\geqslant 0}$ is adaptive submodular w.r.t. distribution $p(\phi)$ if the conditional expected marginal gain of any node does not increase as more nodes are chosen and their states are observed. That is to say, F is adaptive submodular if for all ψ and ψ' such that $dom(\psi) \subseteq dom(\psi')$ and $\psi \sim \psi'$, and for all $v \in V \setminus dom(\psi')$, we have $\Delta(v|\psi) \geqslant \Delta(v|\psi')$.

Based on the above definitions, we show that our utility function $F(S(\pi,\phi),\phi)$ is strong adaptive monotone and adaptive submodular as follows.

Lemma 4.1. *The utility function F is strong adaptive monotone.*

Proof. Given a ψ and $v \notin dom(\psi)$. To show F is strong adaptive monotone, we must show

$$\mathbb{E}[F(dom(\psi),\phi)|\psi \sim \phi] \leqslant \mathbb{E}[F(dom(\psi) \cup \{v\},\phi)|\psi \sim \phi, \phi(v) = o]. \tag{13}$$

Let $N(\psi)$ be the set of nodes that are removed after choosing $dom(\psi)$ and observing ψ. For every $\psi \sim \phi$, the set of removed nodes by selecting $dom(\psi)$ is the same. Let $f(N(\psi))$ be a function denoting the number of edges that are removed after deleting all the nodes in $N(\psi)$. By definition, we have $\mathbb{E}[F(dom(\psi),\phi)|\psi \sim \phi] = f(N(\psi))$. Let ψ' be a partial observation such that $dom(\psi') = dom(\psi) \cup \{v\}$ and $\psi \sim \psi'$. Likewise, we have $\mathbb{E}[F(dom(\psi) \cup \{v\},\phi)|\psi \sim \phi, \phi(v) = o] = f(N(\psi'))$. Clearly, we have $N(\psi) \subseteq N(\psi')$. Since f is a monotone function, we conclude that $f(N(\psi)) \leqslant f(N(\psi'))$. This completes the proof. □

Lemma 4.2. *The utility function F is adaptive submodular.*

Proof. Let ψ and ψ' be two partial observations such that $dom(\psi) \subseteq dom(\psi')$ and $\psi \sim \psi'$. Then, to show F is adaptive submodular, we need to show that $\Delta(u|\psi) \geqslant \Delta(u|\psi')$ holds for all $u \in V \setminus dom(\psi')$.

Without loss of generality, we assume that $dom(\psi) = \{v_1, \ldots, v_i\}$. Let X_j denote the corresponding state of node v_j for $j = 1, \ldots, n$. Then, we partition the set of all the possible states (Ω) into two subsets (Ω_0 and Ω_1) as shown in Table 1.

In the above table, $X(u)$ denotes the state of node u, "$*$" denotes the states that have been observed in ψ, and "$-$" represents the states that have not been observed in ψ. Then, for a possible state of all the nodes ϕ, we have $\Pr[\phi \in \Omega_0|\psi] = 1 - p(u)$ and $\Pr[\phi \in \Omega_1|\psi] = p(u)$, where $p(u)$ denotes the probability of successfully removing node u. Let $\rho_u(dom(\psi),\phi) = F(dom(\psi) \cup \{u\},\phi) - F(dom(\psi),\phi)$. Then, we have

$$\begin{aligned}
\Delta(v|\psi) &= \sum_{\Omega} \Pr[\phi|\psi]\rho_u(dom(\psi),\phi) \\
&= \sum_{\Omega_0} \Pr[\phi|\psi]\rho_u(dom(\psi),\phi) + \sum_{\Omega_1} \Pr[\phi|\psi]\rho_u(dom(\psi),\phi) \\
&= \sum_{\Omega_1} \Pr[\phi|\psi]\rho_u(dom(\psi),\phi) \\
&= \rho_u(dom(\psi),\phi) \sum_{\Omega_1} \Pr[\phi|\psi] \\
&= \rho_u(dom(\psi),\phi)p(u).
\end{aligned} \tag{14}$$

Note that the third equality in the above equation holds due to $\rho_u(dom(\psi),\phi) = 0$ under $\phi \in \Omega_0$. The reason is because, under the Ω_0, we know that node u has not been successfully removed, thereby the corresponding marginal gain is 0. The fourth equality holds because $\rho_u(dom(\psi),\phi)$ is the same for any $\phi \in \Omega_1$. Based on this, we have

$$\Delta(v|\psi) - \Delta(v|\psi') = p(u)(\rho_u(dom(\psi),\phi) - \rho_u(dom(\psi'),\phi)) \geqslant 0,$$

where the last inequality holds because $F(dom(\psi),\phi)$ is submodular for any fixed ϕ. This completes the proof. □

The adaptive greedy algorithm: Here we present an adaptive greedy algorithm for the probabilistic MVC problem. Similar to the greedy algorithm for the MVC problem, the adaptive greedy algorithm works in k rounds. In each round, it chooses the node with the maximal conditional expected marginal gain $\Delta(v|\psi)$ under the current partial observation ψ. We outline the adaptive greedy algorithm in Algorithm 2.

Table 1
States partition.

States	X_1	X_2	$\cdots$	X_i	$X(u)$	X_{i+2}	$\cdots$	X_n	Subsets
Case 0	$*$	$*$	$\cdots$	$*$	0	$-$	$\cdots$	$-$	Ω_0
Case 1	$*$	$*$	$\cdots$	$*$	1	$-$	$\cdots$	$-$	Ω_1

Algorithm 2. The Adaptive Greedy Algorithm.

Input: Network $G = (V, E)$, k, and $p(v)$ for any $v \in V$
Output: A set S with k nodes and δ_k
1: $S \leftarrow \emptyset$, $\psi \leftarrow \emptyset$;
2: **for** iter = 1 to k **do**
3: max $\leftarrow -1$;
4: $Idx \leftarrow 0$;
5: **for** each node $v_i \in (V \setminus S)$ **do**
6: Compute $\Delta(v_i|\psi)$ by Eq. (14)
7: **if** $\Delta(v_i|\psi) >$ max **then**
8: max $\leftarrow \Delta(v_i|\psi)$;
9: $Idx \leftarrow i$;
10: **end if**
11: **end for**
12: $S \leftarrow S \cup \{v_{Idx}\}$;
13: Observe $\phi(v_{Idx})$;
14: $\psi \leftarrow \psi \cup \{(v_{Idx}, \phi(v_{Idx})\}$;
15: **end for**
16: **return** S and $1 - F(dom(\psi), \phi)/m$;

Unlike the greedy algorithm for the MVC problem, in each round, Algorithm 2 needs to observe the state of the selected node (v_{Idx} in line 11). This can be done by tossing a biased coin with probability $p(v_{Idx})$ to determine the state of the selected node (v_{Idx}). In addition, we can use Eq. (14) to compute the conditional expected marginal gain. The time and space complexity of Algorithm 2 are $O(km)$ and $O(m + n)$, respectively.

Theoretically, by using the result derived in [19], Algorithm 2 achieves a $1 - 1/e$ approximate ratio. Formally, we have the following theorem. The proof is similar to the proof presented in [19].

Theorem 4.1. *The node selection policy found by Algorithm 2 obtains $1 - 1/e$ of the value of the optimal policy for the probabilistic MVC problem (Eq. (10)).*

Remark. In [19], Golovin and Krause proposed an algorithm to extend the adaptive greedy algorithm, which is used to solve the adaptive submodular function maximization problem, to handle cost-based model. They proved that their extended algorithm can also produce a $1 - 1/e$ approximation bound. By using their framework, we can easily modify Algorithm 2 to handle the cost-based attack model. Specifically, let c_v be the cost of node v. The generalized algorithm is very similar to Algorithm 2. The only difference is that, in each round, it selects the node with maximal "cost-normalized" expected marginal gain that is $\Delta(v|\psi)/c_v$ for $v \in V$. By using the theory established in [19], such a modified algorithm can achieve a $1 - 1/e$ approximation bound.

5. Experiments

In this section, we conduct extensive experiments on one synthetic and twenty real-world datasets to evaluate the effectiveness and efficiency of our approaches. In the following, we first describe the experimental setup and then report our findings.

5.1. Experimental setup

Datasets: The network datasets used in our experiments are given in Table 2. These networks can be classified into seven categories. (1) Rndtree: the Rndtree dataset is a random tree including 750 nodes and 749 edges. We generate it based on a standard branch process [14]. (2) The co-authorship networks: we collect 5 physics co-authorship networks which are GrQc, Astroph, HepTh, HepPh, and CondMat from Stanford network data collections [31]. These 5 physics co-authorship networks represent the co-authorship over 5 different areas in physics respectively. DBLP (http://www.informatik.uni-trier.de/ley/db/) is a computer science bibliographic dataset. We built a co-authorship graph from a subset of the DBLP data with 78,649 authors. (3) Online social networks (OSNs): we download the Delicious (http://delicious.com/) and Douban (http://www.douban.com/) from ASU social computing data repository [53] and download the Epinions (http://www.epinions.com) and two Slashdot datasets (http://slashdot.org/) from Stanford network data collections [31]. (4) Location-based social networks (LBSNs): the Brightkite and Gowalla are two notable LBSNs. We download these two datasets from Stanford network data collections [31]. (5) Communication networks: we download two Email communication networks (EmailEnron and

Table 2
Summary of the datasets.

Name	#nodes	#edges	Ref.	Description
Rndtree	750	749	[14]	Random tree
GrQc	5242	28,968	[31]	Co-authorship networks
Astroph	18,772	396,100	[31]	
HepTh	9877	51,946	[31]	
HepPh	12,008	236,978	[31]	
CondMat	23,133	186,878	[31]	
DBLP	78,649	382,294	Website	
Delicious	537,392	1,459,778	[53]	Online social networks
Douban	154,908	654,324	[53]	
Epinions	75,872	396,026	[31]	
Slashdot1	77,360	826,544	[31]	
Slashdot2	82,168	867,372	[31]	
Brightkite	58,228	428,156	[31]	Location based social networks
Gowalla	196,591	1,900,654	[31]	
EmailEnron	36,692	367,662	[31]	Communication networks
EmailEuAll	265,182	224,372	[31]	
Gnutella04	10,876	36,308	[31]	P2P networks
Gnutella05	8846	27,572	[31]	
Gnutella06	8717	27,790	[31]	
Gnutella08	6301	18,284	[31]	
NotreDame	325,729	1,522,178	[31]	Web

EmailEuAll) from Stanford network data collections [31]. (6) P2P networks: we employ four P2P networks (Gnutella04, Gnutella05, Gnutella06, and Gnutella08) which are originally collected from Gnutella [31]. (7) Web graphs: we download a web graph dataset from Stanford network data collections [31], which is originally collected from University of Notre Dame.

Parameter settings and experimental environment: There are two parameters in Algorithm 1: the number of hash functions and the size of the bitmap. In all of our experiments, we set the number of hash functions and the size of the bitmap to be 100 and 30, respectively. We conduct our experiments on a Windows Server 2007 with 4xDual-Core Intel Xeon 2.66 GHz CPU, and 8G memory. All the algorithms are implemented in C++.

5.2. Experimental results

Results on trees: Here we conduct experiment on Rndtree dataset. This experiment is designed to evaluate the accuracy of the greedy algorithm described in Section 3.2 and the accuracy of the randomized greedy algorithm given in Algorithm 1. For convenience, we refer to the greedy algorithm and the randomized greedy algorithm as Greedy and RGreedy, respectively. In Section 3.1, we have developed a dynamic programming (DP) algorithm for computing the *k-MVC-impact* and *cumulative k-MVC-impact* of trees optimally. Hence, to evaluate the accuracy of the Greedy and RGreedy, we compare them with the DP algorithm on a random tree dataset. Our results are depicted in Fig. 1.

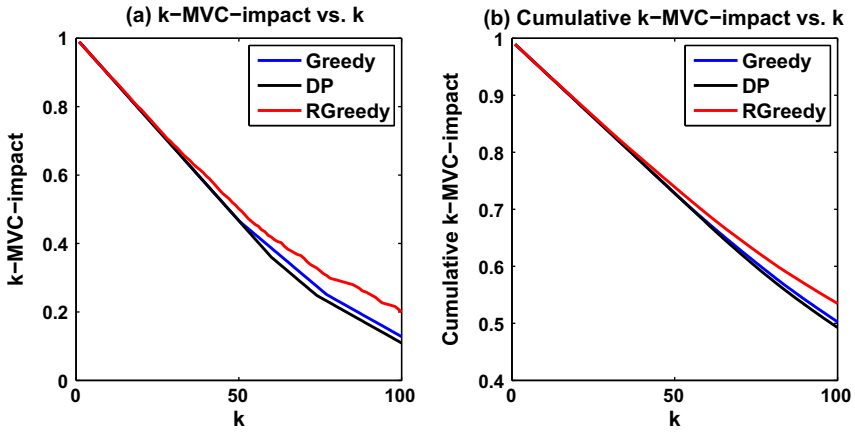

Fig. 1. Result on a random tree.

From Fig. 1(a) and (b), we can clearly see that both the Greedy algorithm and RGreedy algorithm are very close to the optimal DP algorithm. Further, when k is very small (e.g., $k \leqslant 30$), the curves by these three algorithms converge into one curve. The rationale is that both the Greedy and RGreedy achieve near-optimal approximation ratio. In general, both the *k-MVC-impact* and *cumulative k-MVC-impact* decrease as k increases. This is because the impact of MVC attack of a network is inversely proportional to the number of node-attacks.

Results on general networks: Here we conduct experiments on 20 real-world general network datasets. This experiment is designed to measure the impact of MVC attack of these networks. For the directed networks, we consider them as the undirected networks by ignoring the direction of the edges. We use our *k-MVC-impact* and *cumulative k-MVC-impact* as two metrics. Notice that the budget of an attacker, i.e., k, is typically very small in practice. Hence, we mainly focus on measuring the impact of MVC attack of a network under a small budget k. Tables 3 and 4 report the results given $k = 0.1\% \cdot n$ and $0.2\% \cdot n$ respectively, where $n = |V|$. In the following, we concentrate on analyzing the result on $k = 0.1\% \cdot n$ and similar results can be obtained when $k = 0.2\% \cdot n$.

As can be seen in Table 3, the P2P networks are more robust than other types of networks under MVC attack. For example, in the Gnutella05 dataset, after removing 0.1% of nodes, the *k-MVC-impact* and *cumulative k-MVC-impact* by the Greedy algorithm are 0.9838 and 0.9898, respectively. That is to say, there are only 1.62% of edges being deleted after removing 0.1% of nodes in the worst case. This observation indicates that removing a small fraction of peers from the P2P network does not significantly affect the number of links between the peers. Similarly, the co-authorship networks (first 6 rows in Table 3) are shown to be very robust w.r.t. MVC attack. For instance, in the DBLP network, the *k-MVC-impact* and *cumulative k-MVC-impact* by the Greedy algorithm are 0.9618 and 0.9783 after removing 0.1% of nodes, respectively. These results suggest that a small number of "important researchers" leaving the research community will not significantly affect the co-authorship between the scholars. In general, the online social networks (rows 7–11 in Table 3) and the location based social networks (rows 12–13) show poor robustness w.r.t. MVC attack. Taking the Gowalla dataset as an example, the *k-MVC-impact* and the *cumulative k-MVC-impact* by the Greedy algorithm are 0.8329 and 0.8788 when $k = 0.1\% \cdot n$, respectively. In other words, after removing 0.1% of nodes, 16.7% of social ties in the Gowalla network will be deleted. Also, the robustness of the Email communication networks under MVC attack, especially the EmailEuAll network, is very poor. In the EmailEuAll network, the *k-MVC-impact* and *cumulative k-MVC-impact* (for $k = 0.1\% \cdot n$) by the Greedy algorithm are 0.4404 and 0.6408, respectively. In other words, after deleing 0.1% of nodes, the number of residual edges in the EmailEuAll network are only 44.04% of the original edges. This observation suggests that the Email communication networks may be very vulnerable under the MVC attack. In addition, we find that the NotreDame web graph is not very robust w.r.t. the MVC attack, as the *k-MVC-impact* and *cumulative k-MVC-impact* (for $k = 0.1\% \cdot n$) by the Greedy algorithm are 0.8337 and 0.8835, respectively. This result is consistent with the previous results on the "robust yet fragile" nature of the Internet [12], which means that the Internet is robust to random errors but it is vulnerable w.r.t. the intended node attacks. Over all the datasets, we find that the *k-MVC-impact* and *cumulative k-MVC-impact* by the RGreedy algorithm are very close to the *k-MVC-impact* and *cumulative k-MVC-impact* by the Greedy algorithm, respectively. More specifically, for the *k-MVC-impact* and *cumulative k-MVC-impact* when $k = 0.1\% \cdot n$, the maximal absolute differences between the RGreedy algorithm and the Greedy algorithm are only

Table 3
Results on general networks for $k = 0.1\% \cdot n$.

Datasets	*k-MVC-impact*		*Cumulative k-MVC-impact*	
	Greedy	RGreedy	Greedy	RGreedy
GrQc	0.9743	0.9747	0.9841	0.9841
Astroph	0.9660	0.9675	0.9807	0.9811
HepTh	0.9791	0.9794	0.9879	0.9881
HepPh	0.9554	0.9561	0.9751	0.9754
CondMat	0.9641	0.9652	0.9785	0.9787
DBLP	0.9618	0.9643	0.9783	0.9795
Delicious	0.7390	0.7842	0.8220	0.8422
Douban	0.9345	0.9403	0.9625	0.9651
Epinions	0.8515	0.8589	0.9056	0.9088
Slashdot1	0.8825	0.8903	0.9246	0.9286
Slashdot2	0.8800	0.8862	0.9230	0.9259
Brightkite	0.8982	0.9025	0.9353	0.9368
Gowalla	0.8329	0.8401	0.8788	0.8863
EmailEnron	0.8474	0.8529	0.9061	0.9083
EmailEuAll	0.4404	0.4809	0.6408	0.6711
Gnutella04	0.9826	0.9829	0.9888	0.9889
Gnutella05	0.9838	0.9839	0.9898	0.9898
Gnutella06	0.9829	0.9833	0.9893	0.9894
Gnutella08	0.9835	0.9835	0.9894	0.9894
NotreDame	0.8337	0.8492	0.8835	0.8920

Table 4
Results on general networks for $k = 0.2\% \cdot n$.

Datasets	*k-MVC-impact*		*Cumulative k-MVC-impact*	
	Greedy	RGreedy	Greedy	RGreedy
GrQc	0.9536	0.9551	0.9729	0.9732
Astroph	0.9430	0.9449	0.9670	0.9680
HepTh	0.9620	0.9635	0.9787	0.9791
HepPh	0.9176	0.9203	0.9547	0.9559
CondMat	0.9414	0.9442	0.9652	0.9662
DBLP	0.9369	0.9427	0.9636	0.9664
Delicious	0.6512	0.7011	0.7567	0.8064
Douban	0.8924	0.9021	0.9375	0.9433
Epinions	0.7789	0.7948	0.8592	0.8667
Slashdot1	0.8223	0.8389	0.8873	0.8949
Slashdot2	0.8203	0.8336	0.8856	0.8919
Brightkite	0.8479	0.8568	0.9034	0.9074
Gowalla	0.7835	0.8002	0.8426	0.8566
EmailEnron	0.7741	0.7836	0.8575	0.8621
EmailEuAll	0.2755	0.3078	0.4924	0.5272
Gnutella04	0.9720	0.9731	0.9827	0.9831
Gnutella05	0.9733	0.9743	0.9838	0.9841
Gnutella06	0.9739	0.9751	0.9839	0.9843
Gnutella08	0.9701	0.9706	0.9820	0.9821
NotreDame	0.7761	0.8055	0.8437	0.8574

0.0452 (appearing in the Delicious dataset) and 0.0303 (appearing in the EmailEuAll dataset) over all the datasets, respectively. When $k = 0.2\% \cdot n$, the maximal absolute differences for the *k-MVC-impact* and *cumulative k-MVC-impact* are 0.0499 and 0.0497 (both appearing in the Delicious dataset), respectively. These results imply that our RGreedy algorithm is as effective as the Greedy algorithm. In the following, we will systematically study the performance of the RGreedy algorithm.

Performance of the randomized greedy algorithm: In this experiment, we evaluate the effectiveness and efficiency of Algorithm 1, i.e., the RGreedy algorithm. To this end, we compare it with four baseline algorithms, i.e., Greedy, Degree, RCDegree, and Random. Here the Degree algorithm is to select top-k maximal degree nodes, and the Random algorithm is to pick k nodes randomly. The RCDegree algorithm is first to remove top-k' ($k' < k$) high degree nodes and recalculates the degrees of the residual graph, and then recursively removes nodes based on the re-calculated degrees. In our experiments, we set $k' = 0.2 \times k$, and similar results can be observed by other k'. We make use of the *k-MVC-impact* and *cumulative k-MVC-impact* as two metrics to evaluate the effectiveness of the algorithms. Notice that these two metrics inversely measure the effectiveness of the algorithms. Fig. 2 illustrates the effectiveness of the RGreedy algorithm on GrQc dataset. Similar results can be observed on other datasets. From Fig. 2(a) and (b), we can observe that the Greedy algorithm outperforms all other algorithms according to both the *k-MVC-impact* and *cumulative k-MVC-impact* metrics, followed by the RGreedy algorithm, the RCDegree algorithm, the Degree algorithm, and the Random algorithm. Also, we can see that the performance of the RGreedy algorithm is very close to that of the Greedy algorithm, which confirms our theoretical analysis in Section 3.2. In

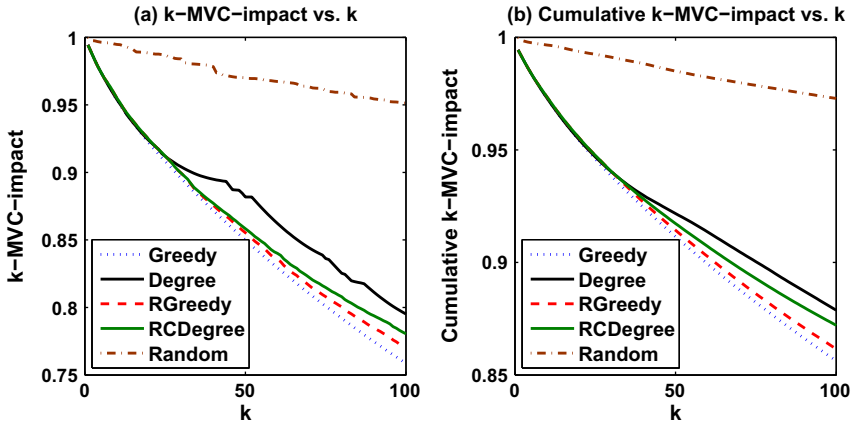

Fig. 2. Effectiveness of the randomized greedy algorithm on GrQc dataset.

addition, we can see that the RCDegree is better than the Degree algorithm. When k is small, the performance of the RCDegree algorithm is close to the performance of the RGreedy algorithm. However, when k increases, the gap between the RCDegree algorithm and our RGreedy algorithm increases, which suggests that the RCDegree algorithm is not very effective when k is large. The reason is because the RCDegree algorithm is without any performance guarantee, while the RGreedy algorithm can achieve a $1 - 1/e - \epsilon$ approximation factor. Also, it is worth mentioning that the effectiveness of the Random algorithm is quite poor.

To evaluate the efficiency of the RGreedy algorithm, we compare the running time of different algorithms. Table 5 shows the running time of various algorithms for computing *k-MVC-impact* over all the datasets used. Here we set $k = 100$ over all the datasets, and similar results can be observed for other k. Since the running time of different algorithms for calculating the *cumulative k-MVC-impact* is very close to computing the *k-MVC-impact*, we omit it for brevity. From Table 5, we find that the RGreedy algorithm is significantly more efficient than the Greedy algorithm. On average, the RGreedy algorithm reduces the running time over the Greedy algorithm by 662%. The most efficient algorithm is the Random algorithm, because it only needs to randomly select k nodes. The RCDegree algorithm is more efficient than the Greedy algorithm, but it is still very expensive when the graph size is very large. Also, we observe that the Degree algorithm is more efficient than the RGreedy and the Greedy algorithms. The reason is that the Degree algorithm can be implemented by a linear-time sorting algorithm, which is $O(n)$ time complexity. These results are consistent with our complexity analysis presented in Section 3.2.

Performance of the adaptive greedy algorithm: In this experiment, we first evaluate the performance of the adaptive greedy algorithm, i.e., Algorithm 2, and then we measure the impact of probabilistic MVC attack of 20 real-world networks based on the adaptive metrics. To this end, for each dataset in Table 2, we generate two types of probabilistic networks, namely homogeneous probabilistic network and heterogeneous probabilistic network, according to two different rules. For the homogeneous probabilistic network, we assign the same probability p for every node. In our experiments, we set p to 0.5, and similar results can be observed for other p. For the heterogeneous probabilistic network, we adopt two probabilistic models. The first model is the power-law model, $p(v) = d_v^{-\alpha}$, where d_v denotes the degree of node v and α is a parameter. In our experiments, we set $\alpha = 1$ and $\alpha = 0.5$ respectively. This power-law model implies that the node with large degree will be robust to attack. This is because the node with large degree is typically an important node, thereby in practice it is built in a more robust way than other nodes. The second model is the uniform distribution model where $p(v)$ is a [0, 1] uniform random variable.

After generating the probabilistic networks, we are now ready to test the performance of our adaptive greedy algorithm. Recall that getting the optimal policy of the probabilistic MVC problem would require enumeration of $C_n^k 2^k$ possible states, which is apparently impractical. Hence, similar to the evaluation methodology introduced in [19], we compare Algorithm 2 to a data dependent bound on the optimal value. Specifically, we develop the adaptive bound as follows. Let $\emptyset = \psi_0 \subset \psi_1 \subset \cdots \subset \psi_k$ be the partial observations observed by Algorithm 2. Let $OPT(\psi_i) = \max_{\pi} \mathbb{E}[F(S(\pi, \phi), \phi)|\psi_i \sim \phi]$ be the optimal value after observing ψ_i. Then, by a similar derivation as in [19], we can obtain $k + 1$ bounds $b_0, b_1, \ldots, b_k$, where $b_i = \mathbb{E}[F(dom(\psi_i), \phi)|\psi_i \sim \phi] + \max_{S \subseteq V, |S| \leqslant k} \sum_{v \in S} \Delta(v|\psi_i)$. Notice that here we use the property of strong adaptive monotone of our utility function F (Lemma 4.1) to derive this bound. After taking the expectation over ϕ, for any π and any i we have

$$\mathbb{E}[F(S(\pi, \phi), \phi)|\psi_i \sim \phi] \leqslant \mathbb{E}[OPT(\psi_i)] \leqslant \mathbb{E}[b_i].$$

Table 5
Running time of different algorithms.

Time (ms)	Greedy	RGreedy	RCDegree	Degree	Random
GrQc	162	37	88	10	10
Astroph	2099	103	1100	19	11
HepTh	302	59	201	12	10
HepPh	1194	71	750	15	10
CondMat	1047	123	653	18	13
DBLP	2459	395	1552	57	11
Delicious	11,858	2664	8023	437	11
Douban	4199	766	2451	110	11
Epinions	2294	379	1450	54	11
Slashdot1	4815	400	2742	59	11
Slashdot2	4974	410	2610	63	11
Brightkite	2507	291	1429	44	10
Gowalla	12,727	970	8750	151	11
EmailEnron	1751	192	781	30	9
EmailEuAll	1846	1308	826	200	10
Gnutella04	230	62	105	13	10
Gnutella05	178	55	93	12	11
Gnutella06	177	52	89	12	10
Gnutella08	118	43	52	10	10
NotreDame	10,354	1609	5879	246	10

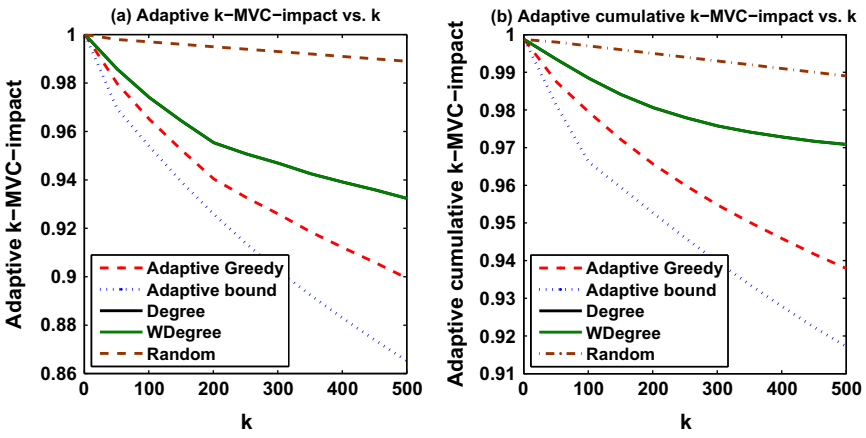

Fig. 3. Performance of the adaptive greedy algorithm on GrQc datasest (Homogeneous probabilistic network, $p = 0.5$ for all nodes).

As a result, for any i, the expectation of b_i is an upper bound on the optimal value of any policy. Analogous to [19], we use the average bound $\bar{b}_i = (\sum_{j=0}^{i} b_j)/(i+1)$ to derive our adaptive bound. That is, for *adaptive k-MVC-impact* and *adaptive cumulative k-MVC-impact*, our adaptive bounds are $\beta_i = 1 - \bar{b}_i/m$ and $\bar{\beta}_i = (\sum_{j=1}^{i} \beta_j)/i$ for $i = 1, \ldots, k$, respectively. In addition, we also compare our algorithm with three other baselines: the Random algorithm, the Degree algorithm, and the WDegree algorithm. The Random algorithm randomly selects nodes to attack, while the Degree algorithm selects nodes to attack according to a decreasing order of their degrees. The WDegree algorithm is the probability weighted Degree algorithm, where the algorithm selects nodes based on the probability weighted degrees (i.e., $p_v \times d_v$). We run all the algorithms 100 times on each dataset, and all the experimental results reported in this subsection are the average results over 100 experiments.

Fig. 3 shows the performance of the adaptive greedy algorithm on GrQc dataset with homogeneous probabilities. Similar results can be observed on other datasets. From Fig. 3, we can clearly see that the performance of the adaptive greedy algorithm is better than the performance of the other baseline algorithms. Note that in the homogeneous probabilistic network, the Degree algorithm and the WDegree algorithm are equivalent. In addition, we can observe that the performance of the adaptive greedy algorithm is close to the adaptive bound. We emphasize that the unknown optimal solution lies between the solution by the adaptive greedy algorithm (red[1] dashed curve) and the adaptive bound (blue dot curve). This result indicates that our adaptive greedy algorithm is very effective and the adaptive bound is tight, which is consistent with our theoretical analysis in Section 4.1.

Figs. 4–6 show the results observed from the heterogeneous probabilistic networks. Similarly, we can see that our solution by the adaptive greedy algorithm is significantly better than the other baselines, and it is also very close to the adaptive bound. This result further confirms that our adaptive greedy algorithm is very effective. In addition, it is worth mentioning that the WDegree algorithm performs much better than the other two baselines in the heterogeneous probabilistic network with uniform distribution, but it still significantly worse than our adaptive greedy algorithm. The reason is because all the baselines algorithm are without any performance guarantee, while our algorithm can achieve a $1 - 1/e$ approximation factor.

The above experiment shows that the adaptive greedy algorithm is very effective to approximately calculate the optimal solution of the probabilistic MVC problem. Here we measure the impact of probabilistic MVC attack in 20 real-world networks using the adaptive metrics (Definitions 4.1 and 4.2), which are calculated by the adaptive greedy algorithm (Algorithm 2). Our results are depicted in Tables 6 and 7 for $k = 0.1\% \cdot n$ and $k = 0.2\% \cdot n$, respectively. To avoid redundancy, we focus on discussing the results for $k = 0.1\% \cdot n$ (Table 6) and similar results can be obtained for $k = 0.2\% \cdot n$. For the heterogeneous probabilistic networks, we focus on describing the results for the power-law distribution with parameter $\alpha = -1$, and similar conclusions can be made for the other models. From Table 6, we can see that, in the homogeneous probabilistic networks, the results of *adaptive k-MVC-impact* and *adaptive cumulative k-MVC-impact* are very similar to the results of the *k-MVC-impact* and *cumulative k-MVC-impact* shown in Table 3, respectively. More specifically, we can find that the P2P networks and co-authorship networks exhibit high robustness to the probabilistic MVC attack, while the communication networks show a low robustness (for example, the EmailEuAll network, row 15 in Table 6). It is worth noting that the values in Table 6 is typically larger than the values shown in Table 3. This is because in the homogeneous probabilistic networks, the node is removed with a constant probability (smaller than 1), while in the case of computing the *k-MVC-impact* metric, the node is removed deterministically. In the heterogeneous probabilistic networks, where the node removal probability is inversely proportional to the degree of the node, we can observe that all the networks exhibit high robustness. The reason is that in such networks, the high degree nodes are very hard to be attacked. Our algorithm selects the node with maximal conditional expected marginal gain to attack, which is typically not a high degree node. On the other hand, the selected node is removed

[1] For interpretation of color in Fig. 1, the reader is referred to the web version of this article.

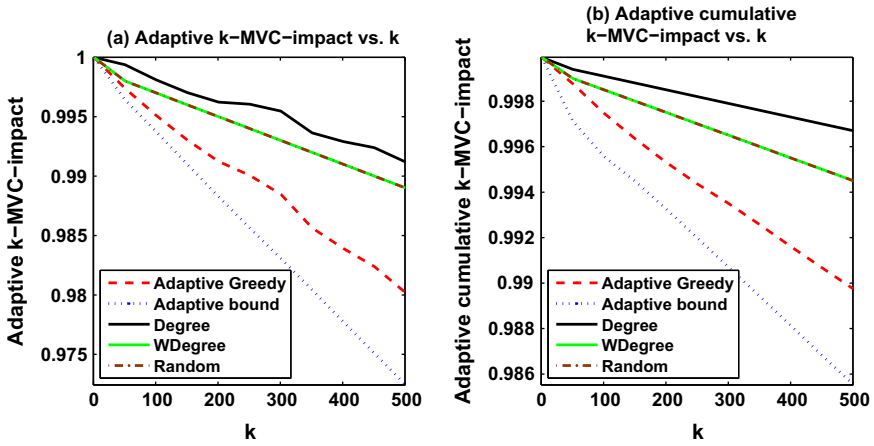

Fig. 4. Performance of the adaptive greedy algorithm on GrQc datasest (Heterogeneous probabilistic network with power-law distribution, $p(v) = d_v^{-1}$ for every node v).

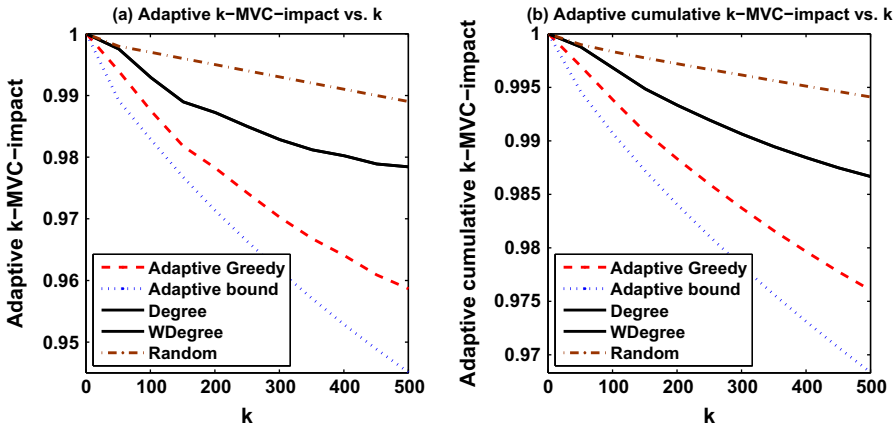

Fig. 5. Performance of the adaptive greedy algorithm on GrQc datasest (Heterogeneous probabilistic network with power-law distribution, $p(v) = d_v^{-1/2}$ for every node v).

by our algorithm according to a probability, which is inversely proportional to its degree. As a result, the algorithm will delete a small number of nodes with relatively low degree, thus deletes a very small number of edges. These results indicate that if the high degree nodes in a network are built in a robust way, then the network is very robust w.r.t. the probabilistic MVC attack.

6. Related work

Robustness of complex networks: Our work is closely related to measure the robustness of complex network. After the seminal work by Albert et al. [1], measuring robustness of complex networks has attracted much attention in the last decade. In order to characterize the robustness of a complex network, most previous studies focused on studying some statistical properties of the network after removing a fraction of nodes or edges. The widely-used statistical properties of a network include the diameter, the size of the giant component, and the average path length. Most subsequent works on network robustness such as [9,8,10,36,30,50] followed this framework. However, all of these works aimed at deriving analytical solutions based on random graph model or some specific graph models, such as star graph model. More recently, Schneider et al. [45] proposed a robustness index based on the size of the giant component and they also showed that the onion-like networks have good robustness according to their index. However, they did not provide a detailed algorithm for computing their index, and the complexity for calculating their index is unknown. Malliaros et al. [40] proposed a robustness metric based on a concept of *expansion*. They also presented an efficient algorithm for computing the metric. Tanaka et al. [46] proposed a dynamical robustness metric to measure the ability of a network to maintain its dynamical activity against node

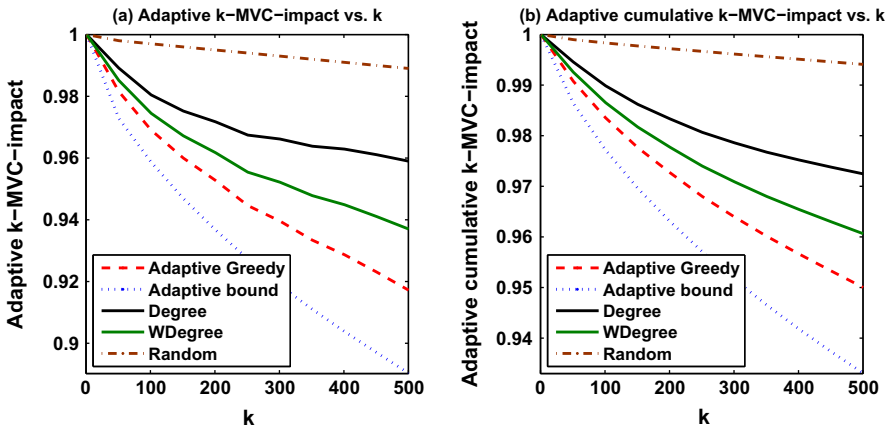

Fig. 6. Performance of the adaptive greedy algorithm on GrQc datasest (Heterogeneous probabilistic network with uniform distribution, $p(v) = U(0,1)$ for every node v).

Table 6
Probabilistic MVC attack on general networks for $k = 0.1\% \cdot n$. Here we use the abbreviation for a certain words, "Adaptive" to "Ada.", "cumulative" to "cum.", "Homogeneous" to "Homo.", and "Heterogeneous" to "Heter".

Datasets	Ada. *k-MVC-impact*		Ada. cum. *k-MVC-impact*	
	Homo.	Heter.	Homo.	Heter.
GrQc	0.9919	0.9998	0.9935	0.9998
Astroph	0.9922	0.9999	0.9943	0.9999
HepTh	0.9924	0.9999	0.9947	0.9999
HepPh	0.9866	0.9999	0.9907	0.9999
CondMat	0.9885	0.9999	0.9926	0.9999
DBLP	0.9905	0.9997	0.9949	0.9999
Delicious	0.9204	0.9893	0.9432	0.9897
Douban	0.9834	0.9896	0.9898	0.9990
Epinions	0.9535	0.9898	0.9698	0.9899
Slashdot1	0.9672	0.9898	0.9771	0.9899
Slashdot2	0.9632	0.9898	0.9750	0.9899
Brightkite	0.9753	0.9898	0.9842	0.9899
Gowalla	0.9550	0.9898	0.9682	0.9899
EmailEnron	0.9512	0.9899	0.9677	0.9899
EmailEuAll	0.8351	0.9779	0.8956	0.9789
Gnutella04	0.9946	0.9998	0.9966	0.9999
Gnutella05	0.9956	0.9996	0.9972	0.9998
Gnutella06	0.9949	0.9999	0.9967	0.9999
Gnutella08	0.9958	0.9997	0.9971	0.9998
NotreDame	0.9566	0.9896	0.9701	0.9898

faults. Subsequently, the same authors also studied how the deletion of hub nodes affects the network dynamics [44]. Unlike these studies, we investigate the impact of a network from an attacker perspective, in which we show how a network is affected by MVC attack.

Another line of research on measuring robustness of complex network is based on network connectivity [18]. Such robustness measures include algebraic connectivity [15], super connectivity [3], conditional connectivity [22], and isoperimetric number [41]. Recently, Wu et al. [52] proposed a spectral measure of robustness of complex network by counting the number of closed walk. Both the connectivity based and spectral robustness measures only consider the topological structure of the network, ignoring the concrete attack models. This may result in some networks with large robustness in terms of these measures, but they are easily attacked by intended attacks. Complementarily, there also exist some studies on vulnerability of complex networks. For instance, in [23], Holme et al. studied the response of complex networks subject to nodes or edges attacks, and they defined the vulnerability of a network by a measure of the average inverse geodesic length. Subsequently, in [6], Boccaletti et al. presented a multi-scale vulnerability measure based on the link betweenness. Recently, Tong et al. [48] proposed a vulnerability measure based on the dominant eigenvalue of the adjacency matrix of the network. Besides network robustness, robustness measures were also studied in the domain of fuzzy reasoning [37]. For example, Li

Table 7
Probabilistic MVC attack on general networks for $k = 0.2\% \cdot n$.

Datasets	Ada. *k-MVC-impact*		Ada. cum. *k-MVC-impact*	
	Homo.	Heter.	Homo.	Heter.
GrQc	0.9919	0.9996	0.9927	0.9997
Astroph	0.9839	0.9999	0.9906	0.9999
HepTh	0.9863	0.9996	0.9917	0.9998
HepPh	0.9786	0.9999	0.9865	0.9999
CondMat	0.9830	0.9998	0.9891	0.9999
DBLP	0.9813	0.9995	0.9902	0.9997
Delicious	0.9003	0.9885	0.9233	0.9889
Douban	0.9710	0.9891	0.9834	0.9896
Epinions	0.9347	0.9895	0.9567	0.9897
Slashdot1	0.9508	0.9897	0.9676	0.9898
Slashdot2	0.9460	0.9897	0.9646	0.9898
Brightkite	0.9608	0.9896	0.9755	0.9898
Gowalla	0.9398	0.9897	0.9573	0.9898
EmailEnron	0.9288	0.9898	0.9526	0.9899
EmailEuAll	0.7865	0.9759	0.8541	0.9780
Gnutella04	0.9916	0.9992	0.9946	0.9996
Gnutella05	0.9932	0.9990	0.9955	0.9995
Gnutella06	0.9927	0.9994	0.9951	0.9997
Gnutella08	0.9919	0.9993	0.9949	0.9996
NotreDame	0.9283	0.9894	0.9339	0.9897

et al. [37] studied several robustness measures of fuzzy connectives. Subsequently, they also investigated the relation between the robustness of fuzzy reasoning and fuzzy implication operators [24].

The preliminary version of this work appears in [35]. In this paper, we revise the preliminary version and make the following extensions. First, we present a polynomial algorithm (the DP algorithm) to compute the impact of MVC attack on trees exactly (Section 3.1). Second, we discuss how to generalize the proposed algorithms to the cost-based attack model (Section 3.3). Third, we study a probabilistic MVC attack problem on general networks, and we present two new metrics based on it (Section 4). Finally, we conduct extensive experiments to support the newly added materials (Section 5).

Submodular and adaptive submodular optimization: Our work is also related to the submodular set function maximization problem [42]. In general, the submodular set function maximization problem is NP-complete. There is a well-known greedy algorithm with $1 - 1/e$ approximate ratio for solving this problem [42]. In practice, many applications have been formulated as the submodular function maximization problem. Such applications include the maximal k coverage problem [49], the influence maximization problem in social networks [25], the observation selection and sensor placement problem [26,27], the document summarization problem [38,39], the spread of misinformation problem [7], and the diversified ranking problem [33,47,34]. All of these problems can be approximately solved by the greedy algorithm described in [42]. In the present paper, we study the MVC problem and present a randomized greedy algorithm for solving it efficiently. Adaptive submodular function generalizes submodular set function to adaptive policies [19]. In [19], Golovin et al. showed that the property of adaptive submodularity results in an efficient greedy algorithm for finding the near-optimal policy. The adaptive submodularity has a number of applications, such as stochastic optimization [19], and active learning [19,4,5]. In the present paper, we prove that the objective function of our probabilistic MVC problem is adaptive submodular, and we develop an efficient greedy algorithm with near-optimal performance guarantee for such a problem.

7. Conclusions

In this paper, we present a study of measuring the impact of MVC attack in complex networks. We define two metrics called *k-MVC-impact* and *cumulative k-MVC-impact*, and propose a dynamic programming algorithm to compute the proposed metrics on trees optimally. We then develop a near-optimal randomized greedy algorithm by using the FM sketch for calculating the proposed metrics on general networks. In addition, we present two adaptive metrics to measure the impact of a network under probabilistic MVC attack, and design a near-optimal adaptive greedy algorithm for computing such metrics. Finally, we extensively evaluate the proposed approaches on 20 real datasets. The results confirm the effectiveness and efficiency of the proposed methods.

There are several future directions deserving further investigation. First, our current metrics are defined on unweighted networks, thereby a direct future direction is to generalize our metrics and techniques to the weighted networks. Second, the proposed metrics are based on a natural utility function, i.e., the number of edges. It would be interesting to define new metrics based on other utility functions, such as the number of triangles, the number of closed paths, and so on. In addition, our metrics are based on the node attack models. One can also define new metrics based on the edge attack models, in which the

problem is to delete a set of edges so as to maximize (or minimize) some predefined utility functions. Finally, it would also be interesting to apply our metrics and algorithms to real-world applications.

Acknowledgements

The work was supported by grants GRF 418512, 411211, and 411310 from HK-RGC.

References

[1] R. Albert, H. Jeong, A.L. Barabsi, Error and attack tolerance of complex networks, Nature 406 (2000).
[2] A.L. Barabási, R. Albert, Emergence of scaling in random networks, Science (1999).
[3] D. Bauer, F. Boesch, C. Suffel, R. Tindell, Connectivity extremal problems and the design of reliable probabilistic networks, in: Theory and application of graphs, 1981.
[4] G. Bellala, S.K. Bhavnani, C. Scott, Extensions of generalized binary search to group identification and exponential costs, in: NIPS, 2010.
[5] G. Bellala, S.K. Bhavnani, C. Scott, Group-based active query selection for rapid diagnosis in time-critical situations, IEEE Trans. Inform. Theor. (2012).
[6] S. Boccaletti, J. Buldu, R. Criado, J. Flores, V. Latora, J. Pello, M. Romance, Multiscale vulnerability of complex network, Chaos (2007).
[7] C. Budak, D. Agrawal, A. El Abbadi, Limiting the spread of misinformation in social networks, in: WWW, 2011.
[8] D.S. Callaway, M.E.J. Newman, S.H. Strogatz, D.J. Watts, Network robustness and fragility: percolation on random graphs, Phys. Rev. Lett. 85 (25) (2000).
[9] R. Cohen, K. Erez, D. ben Avraham, S. Havlin, Resilience of the internet to random breakdowns, Phys. Rev. Lett. 85 (21) (2000).
[10] R. Cohen, K. Erez, D. ben Avraham, S. Havlin, Breakdown of the internet under intentional attack, Phys. Rev. Lett. 86 (16) (2001).
[11] G. Cornuejols, G.L. Nemhauser, L.A. Wolsey, Worst-case and probabilistic analysis of algorithms for a location problem, Oper. Res. 28 (4) (1980) 847–858.
[12] J.C. Doyle, D.L. Alderson, L. Li, S. Low, M. Roughan, S. Shalunov, R. Tanaka, W. Willinger, The robust yet fragile nature of the internet, PNAS 102 (41) (2005) 14497–14502.
[13] M. Durand, P. Flajolet, Loglog counting of large cardinalities (extended abstract), in: ESA, 2003, pp. 605–617.
[14] R. Durrett, Random Graph Dynamics, Cambridge University Press, 2006.
[15] M. Fiedler, Algebraic connectivity of graphs, Czech. Math. J. 23 (98) (1973) 298–305.
[16] P. Flajolet, E. Fusy, O. Gandouet, F. Meunier, Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm, in: ESA, 2003, pp. 605–617.
[17] P. Flajolet, G.N. Martin, Probabilistic counting algorithms for data base applications, J. Comput. Syst. Sci. 31 (2) (1985) 182–209.
[18] H. Frank, I.T. Frisch, Analysis and design of survivable network, IEEE Trans. Commun. Technol. 18 (5) (1970) 501–519.
[19] D. Golovin, A. Krause, Adaptive submodularity: theory and applications in active learning and stochastic optimization, J. Artif. Intell. Res. (JAIR) 42 (2011) 427–486.
[20] P.R. Goundan, A.S. Schulz, Revisiting the Greedy Approach to Submodular Set Function Maximization, Technical Report, MIT, 2008.
[21] Q. Han, Y. Ye, H. Zhang, J. Zhang, On approximation of max-vertex-cover, Eur. J. Oper. Res. 143 (2) (2002) 342–355.
[22] F. Harary, Conditional connectivity, Networks 13 (1983) 346–357.
[23] P. Holme, B.J. Kim, C.N. Yoon, S.K. Han, Attack vulnerability of complex networks, Phys. Rev. Lett. (2002).
[24] J. Jin, Y. Li, C. Li, Robustness of fuzzy reasoning via logically equivalence measure, Inf. Sci. 15 (2007) 5103–5117.
[25] D. Kempe, J.M. Kleinberg, É. Tardos, Maximizing the spread of influence through a social network, in: KDD, 2003.
[26] A. Krause, C. Guestrin, Near-optimal observation selection using submodular functions, in: AAAI, 2007, pp. 1650–1654.
[27] A. Krause, A.P. Singh, C. Guestrin, Near-optimal sensor placements in gaussian processes: theory, efficient algorithms and empirical studies, J. Mach. Learn. Res. 9 (2008) 235–284.
[28] R. Kumar, K. Punera, A. Tomkins, Hierarchical topic segmentation of websites, in: KDD, 2006.
[29] T. Lappas, E. Terzi, D. Gunopulos, H. Mannila, Finding effectors in social networks, in: KDD, 2010.
[30] S. Latifi, E. Saberinia, X. Wu, Robustness of star graph network under link failure, Inf. Sci. 178 (3) (2008) 802–806.
[31] J. Leskovec, Standford Network Analysis Project, 2010. <http://snap.standford.edu>.
[32] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J.M. VanBriesen, N.S. Glance, Cost-effective outbreak detection in networks, in: KDD, 2007.
[33] R.H. Li, J.X. Yu, Scalable diversified ranking on large graphs, in: ICDM, 2011, pp. 1152–1157.
[34] R.H. Li, J.X. Yu, Scalable diversified ranking on large graphs, IEEE Trans. Knowl. Data Eng. (2013).
[35] R.H. Li, J.X. Yu, X. Huang, H. Cheng, Z. Shang, Measuring robustness of complex networks under MVC attack, in: CIKM, 2012.
[36] T.K. Li, J.J.M. Tan, L.H. Hsu, Hyper hamiltonian laceability on edge fault star graph, Inf. Sci. 165 (2004) 59–71.
[37] Y. Li, D. Li, W. Pedrycz, J. Wu, An approach to measure the robustness of fuzzy reasoning, Int. J. Intell. Syst. 20 (4) (2005) 393–413.
[38] H. Lin, J. Bilmes, Multi-document summarization via budgeted maximization of submodular functions, in: HLT-NAACL, 2010.
[39] H. Lin, J. Bilmes, A class of submodular functions for document summarization, in: ACL, 2011.
[40] F.D. Malliaros, V. Megalooikonomou, C. Faloutsos, Fast robustness estimation in large social graphs: communities and anomaly detection, in: SDM, 2012.
[41] B. Mohar, Isoperimetric number of graphs, J. Combin. Theor. Ser. B 47 (3) (1989) 274–291.
[42] G.L. Nemhauser, L.A. Wolsey, M.L. Fisher, An analysis of approximations for maximizing submodular set functions-i, Math. Program. 14 (1978) 265–294.
[43] C.R. Palmer, P.B. Gibbons, C. Faloutsos, ANF: a fast and scalable tool for data mining in massive graphs, in: KDD, 2002, pp. 81–90.
[44] R. Quax, A. Apolloni, P.M.A. Sloot, The diminishing role of hubs in dynamical processes on complex networks, J.R. Soc. Interface 10 (2013).
[45] C.M. Schneider, A.A. Moreira, S. Jos, J. Andrade, S. Havlin, H.J. Herrmann, Mitigation of malicious attacks on networks, PNAS 108 (10) (2011) 427–486.
[46] G. Tanaka, K. Morino, K. Aihara, Dynamical robustness in complex networks: the crucial role of low-degree nodes, Sci. Rep. 2 (232) (2012).
[47] H. Tong, J. He, Z. Wen, R. Konuru, C.Y. Lin, Diversified ranking on large graphs: an optimization viewpoint, in: KDD, 2011.
[48] H. Tong, B.A. Prakash, C.E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, D.H. Chau, On the vulnerability of large graphs, in: ICDM, 2010.
[49] V.V. Vazirani, Approximation Algorithms, Springer, 2004.
[50] D. Walker, S. Latifi, Improving bounds on link failure tolerance of the star graph, Inf. Sci. 180 (3) (2010) 2571–2575.
[51] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature (1998).
[52] J. Wu, M. Barahon, Y.J. Tan, H.Z. Deng, Spectral measure of structural robustness in complex networks, IEEE Trans. Syst. Man Cybern. – Part A 41 (6) (2011).
[53] R. Zafarani, H. Liu, Social Computing Data Repository at ASU, 2009. <http://socialcomputing.asu.edu>.