Trapping Malicious Crawlers in Social Networks

Shiju Li
Florida Institute of Technology
sli2015@my.fit.edu

Chul-Ho Lee
Florida Institute of Technology
clee@fit.edu

Do Young Eun
North Carolina State University
dyeun@ncsu.edu

ABSTRACT

In this paper, we study a problem of trapping malicious web crawlers in social networks to minimize the attacks from crawlers with malicious intents to steal personal/private information. The problem is to find where to place a given set of traps over a graph so as to minimize the expected number of users who possibly fall prey to a (possibly random) set of malicious crawlers, each of which traverses the graph in a random-walk fashion for a random finite time. We first show that this problem is NP-hard and also a monotone submodular maximization problem. We then present a greedy algorithm that achieves a $(1-1/e)$-approximation. We also develop an (ϵ, δ)-approximation Monte Carlo estimator to ease the computation of the greedy algorithm and thus make the algorithm scalable for large graphs. We finally present extensive simulation results to show that our algorithm significantly outperforms other baseline algorithms based on various centrality measures.

CCS CONCEPTS

• **Information systems** → **Web crawling**; **Social networks**;

KEYWORDS

social networks, crawlers, random walks, submodular optimization

ACM Reference Format:
Shiju Li, Chul-Ho Lee, and Do Young Eun. 2020. Trapping Malicious Crawlers in Social Networks. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3412004

1 INTRODUCTION

Crawling web graphs and online social networks (OSNs) has been central to building the databases of search engines for serving user queries and also for complex network analysis, i.e., understanding the properties of nodes, their relationships (edges), and more sophisticated relationship among multiple nodes (subgraph patterns) of the large complex networks. In particular, it has been a key enabler for *graph sampling*, which has been extensively studied in the literature, for estimating various target quantities, including the size of a subset of nodes, degree distribution, assortativity coefficient, network-average and global clustering coefficients, and locally connected subgraph patterns such as triples, motifs and graphlets [9, 14, 25, 44, 48].

The crawlers for graph sampling are often implemented in the form of *random walks* to achieve statistical guarantees, while leveraging the public local-neighborhood-only interfaces (APIs) provided by the target-network companies, or following the URLs and hyperlink structures (i.e., HTML screen scraping), especially when the public APIs are not available [14, 31, 37]. The graph sampling via crawling has been a non-trivial problem both in theory and practice [42, 48], since the operation of crawlers can only be valid for a limited amount of time due to (rather stringent) rate limits on the usage of APIs and web access per IP address of a crawler, and thus collected samples can only provide a partial view of the underlying graph and may not be sufficient for estimation and inference.

On the flip side, crawling can be misused and leveraged for malicious purposes. Specifically, the topological information of OSNs can be collected by adversaries who create false profiles, or gain access to legitimate users who are already well-connected to others by compromising their credentials, and then exploit the neighbor-to-neighbor structures. They can be used with malicious intents to identify well-connected individuals and communities of users with common interests [34, 45], to infer private information from users' friends [29, 47], and to facilitate the extraction of user profiles [2, 11]. It is thus crucial to prevent malicious crawlers from compiling a collection of private social information and protect user privacy from such adversarial attacks.

In this paper, we consider a relevant problem, which is to find an optimal set of nodes to place 'traps' over a social graph (modeling a web-based social network or an OSN) in order to minimize the expected number of users who possibly fall prey to malicious crawlers. While the privilege of privacy control is generally given to users to a large extent, many users still fail to protect their private information, thereby becoming vulnerable to the adversaries.[*] To remedy this problem, traps can be implemented in the form of 'spider' traps/honeypots [24, 40] or with anti-crawling measures such as Captchas and user verification, to catch the crawlers and prevent them from further accessing user pages to steal private data. As another possible solution, restrictive 'access control' [2, 4, 50] can be enforced *selectively* to a group of users on behalf of them, which can be viewed as traps, to prohibit the crawlers from prying into the underlying social network (by exploiting its neighbor-to-neighbor structure) to the maximum extent possible. This problem can also be viewed, in a weaker form, as that of finding an optimal set of vantage points over the network for monitoring access patterns and traffic to detect anomalous access behaviors [43, 46].[†]

[*]It is reported in [2] that only 10% of Facebook users remove their profile pictures and friend information from search results.

[†]Enforcing restrictive access control to the entire social network and monitoring over the entire network may be infeasible, due to possible negative effects of such an

We formulate and study the problem as a combinatorial optimization problem of finding a set of nodes on a general social graph to trap a (possibly random) set of malicious crawlers to minimize the expected number of their potential victims. Here each crawler starts from a node chosen uniformly at random from a 'launch' set S, in which nodes are possibly disconnected and scattered. It then independently moves over the graph in a random-walk fashion for a *random finite* time, which can have any arbitrary distribution. We provide the following contributions in this paper.

- First, we present a system model and provide its mathematical analysis to set the stage for the optimization problem.
- We next prove that the problem is NP-hard, but also show that it essentially becomes a monotone submodular maximization problem.
- Leveraging this nice functional property, we propose a greedy algorithm that achieves a $(1-1/e)$-approximation to the optimal solution of the original NP-hard problem. Furthermore, to make the greedy algorithm practical and scalable for large graphs, we provide an (ϵ, δ)-approximation Monte Carlo estimator that enables an efficient computation of the key quantity required in the algorithm operation with provable guarantee.
- Finally, we present extensive simulation results on various real-world graphs to demonstrate the efficacy and scalability of our greedy algorithm. In particular, the simulation results show that our algorithm significantly outperforms other baseline algorithms based on various centrality measures.

2 RELATED WORK

This work is relevant to a rich literature that is concerned about how to manipulate the structure of an underlying graph, i.e., removing nodes and/or edges, to maximally suppress diffusion processes on the graph, ranging from the spread of epidemics to the cascade of influence (or information diffusion), under budget constraints.

First, there is a lot of work for developing immunization strategies, which often boil down to removing a set of nodes (vaccinating) or edges (quarantining), to control the spread of epidemics under the susceptible-infected-susceptible (SIS) and susceptible-infected-recovered (SIR) models on graphs. It was shown by Chakrabarti *et al.* [5] and Ganesh *et al.* [12] that under the SIS/SIR models, an epidemic dies out quickly if the spectral radius of the underlying graph, i.e., the largest eigenvalue of its adjacency matrix, is less than a 'threshold' that depends on the model parameters.[‡] This observation has motivated several studies on the development of strategies to find a set of nodes or edges *for removal* to reduce the spectral radius of the resulting graph below the threshold (or to minimize the spectral radius) so as to ensure the (quick) extinction of an epidemic, e.g., [5, 7, 8, 38, 41]. Other *spectral* measures have also been similarly used to identify critical nodes and/or edges from the standpoint of network connectivity [6, 49]. Furthermore, centrality measures, including the betweenness centrality, have been used as effective means for combating epidemics by immunizing a fraction of nodes or removing edges [13, 39].

In addition, there is another body of literature, which has been based on the diffusion models for the spread of influence, such as the independent cascade (IC) and linear threshold (LT) models, since the seminal work by Kempe *et al.* [19] on the influence maximization problem. Note that these models are still different from the epidemic models, because the former models define node-to-node influence operations as a 'one-time' process, i.e., an active node attempts to influence its neighbor only once, while the latter ones allow node-to-node infections to remain effective. Kimura *et al.* [21] studied the problem of 'minimizing' the propagation of rumors by blocking a limited number of links under the IC model. In [16], the authors tackled the 'influence blocking maximization' problem under the competitive LT model (a variant of the LT model), which can be viewed as a node deletion problem. Furthermore, Kuhlman *et al.* [22] proposed heuristic algorithms for the problem of edge removal under a deterministic variant of the LT model. Khalil *et al.* [20] studied the problems of deleting and adding edges to minimize and maximize the spread of influence, respectively, under the LT model. More recently, Nguyen *et al.* [35] addressed the problem of removing nodes or edges to stop a so-called cyber-epidemics under the LT model.

While the process of random walks is another class of diffusion processes, it is fundamentally different from the above diffusion models, especially from a modeling perspective. The epidemic and influence-diffusion models are characterized by the time-varying behavior of the set of 'infected' nodes and that for 'active' nodes, respectively. The process of random walks, however, is characterized by their trajectories, which are the sequences of nodes visited by walks. Furthermore, in contrast to the aforementioned literature, there are only few studies on manipulating the underlying network structure to facilitate, *but not suppress*, the diffusion of random walks or merely studying relevant hitting times of random walks. Mavroforakis *et al.* [30] studied the so-called k-ARW-centrality problem to place k absorbing nodes on a graph in order to minimize the absorption time of an 'infinite-length' random walk that is a simple random walk with occasional jumps to a set of nodes. In [27], the authors solved the problem of finding a set of nodes to minimize the total hitting time of a 'fixed-length' random walk to the rest of the graph. Golnari *et al.* [15] analyzed the hitting times of random walks when they can avoid or must go through a specific node on a graph, but without considering any optimization problem.

3 SYSTEM MODEL AND PROBLEM FORMULATION

3.1 Problem Setup and Justification

Consider a general connected, undirected graph $G=(N,E)$ with size $|N|=n$, to model a web-based social network or an OSN. The graph G is an *arbitrary* given graph, as long as it is connected and undirected. The graph G is defined by an $n \times n$ adjacency matrix $\mathbf{A}=[A_{ij}]$ with elements $A_{ij}=1$ if there is an edge between nodes i and j, i.e., $(i,j) \in E$, and $A_{ij}=0$ if otherwise. Suppose that time is divided into discrete slots. The problem of interest is to find where to place a given set of *traps* over graph G so as to minimize the expected number of users who fall prey to a set of malicious

enforcement (e.g., some users may lodge complaints and even quit the social network) and the sheer size of the underlying network.

[‡]Specifically, the threshold is the ratio of the recovery rate δ to the infection rate β under the SIS/SIR models, where a susceptible node becomes infected with rate β times the number of infected neighboring nodes, and any infected node is independently recovered with rate δ. Once an infected node becomes healthy, it is again susceptible to infection under the SIS model, whereas it is permanently cured under the SIR model.

crawlers, when each of crawlers traverses G in a random-walk fashion *for a random finite time* that is drawn from *any* arbitrarily given distribution.

The rationale behind the 'random walk' trajectories of malicious crawlers stems from the fact that random walks have been widely used and recommended as the trajectories of crawlers for sampling OSNs and webs with *statistical guarantees* [9, 14, 25, 44, 48]. The 'finite length' also comes from the fact that the usage of APIs and web access per IP address of a crawler are rate-limited [14, 31, 37, 42, 48]. Unlike the graph sampling literature, this work considers the adversarial aspects of web crawling when misused and leveraged for malicious purposes. We note that the trajectories of malicious crawlers may be governed by more complicated (stochastic) processes, and this extension would be an interesting future direction. Nonetheless, as shall be shown later, the problem at hand is still non-trivial while leading to fundamental insights.

In addition, the notion of 'traps' can be interpreted in practice as follows. First, the traps can be *spider traps/honeypots* [24, 40], which are the fake pages or embedded links that are only accessible by (malicious) crawlers, not by humans, and get them stuck in infinite loops or crashing. Second, they can also be implemented with *anti-crawling* measures such as Captchas and user verification, preventing the crawlers from moving forward. Third, the traps can be a group of users' pages where some restrictive access control [2, 4, 50] is enforced on behalf of them to keep their private information such as user profiles and their list of friends *undisclosed*, which in turn prohibits the crawlers from exploiting the neighbor-to-neighbor structure for further crawling. Finally, in a broader sense, they can be viewed as vantage points over the network for monitoring access patterns and traffic to detect anomalous access behaviors and build a blocklist of IP addresses. To summarize, the notion of traps for anti-crawling/scraping has been widely used in practice, and we then look at the problem of *where* those traps need to be installed by mathematically formulating an optimization problem.

3.2 Model Description

Consider K malicious crawlers, each of which has a random length of L_k. While K can be random, for ease of exposition, we assume that K is a fixed value. We set $L_k, k=1,2,\ldots,K$, to be *i.i.d.* copies of L that has any general distribution over a finite support in that $\lim_{l\to\infty}\mathbb{P}\{L=l\}=0$. Each crawler k initially starts from a node s chosen uniformly at random from a given node set $S\subset N$, and independently moves over G for L_k steps in a random-walk fashion. Here, this source-node set S can also be *arbitrary*, i.e., nodes in S do not need to be connected. For crawler k, we then consider the nodes outside S that crawler k visits for L_k steps as *victims*, or the nodes that fall prey to crawler k having malicious intents to steal personal or private information. Thus, the number of victims by crawler k is identical to the number of *unique* nodes visited by k.

We assume that when crawler k moves from a node to one of its neighbors at each step, it only moves to one of the neighbors that are *not* in S, since it has no incentive to visit S from which the crawlers are launched. Thus, the movement of crawler k is modeled as a simple random walk on a *modified* graph, say G_S, instead of the original graph G, where all the incoming edges into S are removed (but the nodes in S still remain in the graph). To be precise, it is defined by $G_S\triangleq(N,E_S)$, where $E_S\triangleq\{(i,j)\in E\mid j\notin S\}$. Note that $A_{ij}=0$ for $j\in S$. The transition probability of the simple random walk from node u to v on G_S is accordingly defined by $P_{uv}=1/d_u$ for $(u,v)\in E_S$ and $P_{uv}=0$ for $(u,v)\notin E_S$, where d_u is the degree of node u on the modified graph G_S, i.e., $d_u=\sum_{v\in N}A_{uv}$.

Let $\mathcal{V}$ be the expected number of nodes that are victimized by K crawlers, where the expectation is taken over the random trajectories of the crawlers and their random lifetimes. For mathematical tractability, we consider $\mathcal{V}$ as the expected value of the sum of the numbers of victims (or unique visits) by K crawlers. In other words, some node i may be visited/victimized by multiple crawlers in which case they are counted separately. Nonetheless, it is still non-trivial to characterize $\mathcal{V}$ analytically and the mathematical analysis serves as a crucial point of departure for the design of a practically usable algorithm. Furthermore, we evaluate our proposed algorithm and demonstrate its superiority over other algorithms through extensive simulations *without* such an assumption in Section 5.

Fix crawler k. Letting $S^c\triangleq N\setminus S$, we first evaluate the probability that node $i\in S^c$ is victimized by crawler k. Define $X_k(t)$ to be the position at time t of the corresponding *infinite-length* simple random walk on G_S, and $T_k(i)$ to be its first hitting time of node i, which is given by $T_k(i)\triangleq\min\{t\geq 0\mid X_k(t)=i\}$. Then, we observe that node i is victimized by crawler k if it is visited by k during its lifetime L_k. In other words, it happens with probability that the first hitting time of node i is no greater than L_k, which is given by

$$\begin{aligned}\mathbb{P}_{I_k}\{T_k(i)\leq L_k\}&\triangleq\mathbb{P}\{T_k(i)\leq L_k\mid X_k(0)\sim\mathbf{u}_S\}\\&=\frac{1}{|S|}\sum_{s\in S}\mathbb{P}\{T_k(i)\leq L_k\mid X_k(0)=s\},\end{aligned}\qquad(1)$$

where $\mathbf{u}_S$ denotes a uniform distribution over the set S, and $I_k\triangleq\{X_k(0)\sim\mathbf{u}_S\}$ denotes an event that the initial position $X_k(0)$ is chosen from S uniformly at random. Note that since L_k is random, $\mathbb{P}_{I_k}\{T_k(i)\leq L_k\}$ is not in a simple form of the CDF of the first hitting time $T_k(i)$, which has been studied extensively in the literature [3, 36]. Nonetheless, we below show that it can also be written as a weighted sum of $\mathbb{P}_{I_k}\{T_k(i)=t\}$, with coefficients given by the CCDF of L_k, and this representation shall be useful in the subsequent analysis.

Lemma 1. *The probability that node i becomes a victim of crawler k, starting from a uniformly random node from S, is given by*

$$\mathbb{P}_{I_k}\{T_k(i)\leq L_k\}=\sum_{t=0}^{\infty}q_t\cdot\mathbb{P}_{I_k}\{T_k(i)=t\},\qquad(2)$$

where $q_t\triangleq\mathbb{P}\{L\geq t\}$, *with* $\lim_{t\to\infty}q_t=0$.

Proof. See our technical report [28]. □

We next define V_k to be the number of victims by crawler k, which is given by

$$V_k\triangleq\sum_{i\in S^c}\mathbb{1}\{T_k(i)\leq L_k\},\qquad(3)$$

where $\mathbb{1}\{A\}$ denotes an indicator function of an event A, having $\mathbb{1}\{A\}=1$ if A occurs, and $\mathbb{1}\{A\}=0$ otherwise. From (3), we have

$$\mathbb{E}[V_k|I_k]=\mathbb{E}\Big[\sum_{i\in S^c}\mathbb{1}\{T_k(i)\leq L_k\}\;\Big|\;I_k\Big]$$

$$= \sum_{i \in S^c} \mathbb{E}\big[\mathbb{1}\{T_k(i) \le L_k\} \mid I_k\big] = \sum_{i \in S^c} \mathbb{P}_{I_k}\{T_k(i) \le L_k\}, \quad (4)$$

where the second equality is from the linearity of conditional expectation. Then we can write $\mathcal{V}$ – the expected number of nodes victimized by K crawlers, when each of them independently starts from node s chosen from S uniformly at random, as

$$\mathcal{V} = \mathbb{E}\Big[\sum_{k=1}^{K} V_k \,\Big|\, I_1, I_2, \ldots, I_K\Big] = \sum_{k=1}^{K} \mathbb{E}[V_k \mid I_k] = K \cdot \mathbb{E}[V_1 \mid I_1], \quad (5)$$

where we have used the linearity of conditional expectation and the fact that V_k depends only on I_k for each k. Therefore, by (2), (4), and (5), we have

$$\mathcal{V} = K \sum_{i \in S^c} \sum_{t < \infty} q_t \cdot \mathbb{P}_I\{T(i) = t\}. \quad (6)$$

Hereafter, we drop the subscript k from I_k and $T_k(i)$ for brevity.

3.3 Problem Statement

We are now interested in finding a set of nodes $W \subset S^c$ of size $|W| = b$, i.e., where to place b 'traps', to catch K crawlers so as to minimize the expected number of their victims $\mathcal{V}$, for a given set S. To properly model the effect of traps W on G_S, we assume that once each crawler is trapped by any node in W, it stays there for the rest of its lifetime. To model this, for any given choice of W, we define the resulting graph $G_W \triangleq (N, E_W)$, where $E_W \triangleq \{(i,j) \in E_S \mid i \notin W\}$, i.e., all the outgoing edges from W are removed from $G_S = (N, E_S)$. Note that the entire node set N still remains intact, while the outgoing edges from W (and also the incoming edges into S) are removed. Furthermore, to properly indicate the differences of our target quantity $\mathcal{V}$ *without and with* W traps, we reserve $\mathcal{V}$ for the former and introduce $\mathcal{V}(W)$ for the latter. To be precise, for a given choice of W and its corresponding graph G_W, we define

$$h_t(s, i; G_W) \triangleq \mathbb{P}\{T(i) = t \,|X(0) = s\}, \quad s \in S, i \in S^c. \quad (7)$$

With a slight abuse of notation, we also define

$$h_t(\mathbf{u}_S, i; G_W) \triangleq \mathbb{P}_I\{T(i) = t\} = \mathbb{P}\{T(i) = t \mid X(0) \sim \mathbf{u}_S\}$$
$$= \frac{1}{|S|} \sum_{s \in S} h_t(s, i; G_W), \quad i \in S^c. \quad (8)$$

Here these definitions are given to clearly indicate the hitting time probabilities to be the ones obtained on G_W. Thus, from (6), we can write

$$\mathcal{V}(W) = K \sum_{i \in S^c} \sum_{t < \infty} q_t \cdot h_t(\mathbf{u}_S, i; G_W). \quad (9)$$

Using the notations in (7)–(8), we can also rewrite $\mathcal{V}$ in (6) as

$$\mathcal{V} = K \sum_{i \in S^c} \sum_{t < \infty} q_t \cdot h_t(\mathbf{u}_S, i; G_S). \quad (10)$$

For a given set S and a given budget (or size) b of W, our optimization problem is then to find

$$\boldsymbol{\mathcal{P}1}: \qquad W^\star = \operatorname*{arg\,min}_{W \subseteq S^c \,:\, |W| = b} \mathcal{V}(W).$$

4 MAIN RESULTS

We present our main results for the optimization problem $\boldsymbol{\mathcal{P}1}$. In particular, we propose a $(1-1/e)$-approximation greedy algorithm for $\boldsymbol{\mathcal{P}1}$, by showing that its equivalent problem is a monotone submodular maximization problem. We also propose an (ϵ, δ)-approximation Monte Carlo (MC) estimator that allows us to efficiently compute the key quantity $\mathcal{V}(W)$ required in the execution of the greedy algorithm, which makes the algorithm practical and scalable for large graphs.

4.1 Submodularity and Greedy Algorithm

We first show that $\boldsymbol{\mathcal{P}1}$ is NP-hard. Specifically, we prove the NP-hardness by reducing the decision problem of the Vertex Cover problem [10] to the decision problem of $\boldsymbol{\mathcal{P}1}$.

Theorem 1. *$\boldsymbol{\mathcal{P}1}$ is NP-hard.*

Proof. See Appendix A. □

It is thus naturally expected that developing a computationally efficient algorithm, even an approximation algorithm, for $\boldsymbol{\mathcal{P}1}$ is simply a non-trivial task. Nonetheless, by transforming $\boldsymbol{\mathcal{P}1}$ to its equivalent problem, we next show that it boils down to the problem of maximizing a monotone submodular function, which is amenable to the development of a greedy algorithm with provable guarantee. To proceed, we collect the following definitions.

Definition 1 (Monotonicity [32]). *A function $f : 2^N \to \mathbb{R}$ is monotone if, for every $A \subseteq B \subseteq N$, $f(A) \le f(B)$.*

Definition 2 (Submodularity [32]). *A function $f : 2^N \to \mathbb{R}$ is submodular if, for every $A \subseteq B \subseteq N$ and every $u \in N \setminus B$, $f(A \cup \{u\}) - f(A) \ge f(B \cup \{u\}) - f(B)$.*

In other words, f is submodular if it satisfies the 'diminishing returns' property, which implies that the marginal gain from adding an element to a set A diminishes with increasing size of A.

We next define

$$\mathcal{D}(W) \triangleq \mathcal{V} - \mathcal{V}(W). \quad (11)$$

Then it is easy to see that $\boldsymbol{\mathcal{P}1}$ becomes equivalent to finding

$$\boldsymbol{\mathcal{P}2}: \qquad W^\star = \operatorname*{arg\,max}_{W \subseteq S^c \,:\, |W| = b} \mathcal{D}(W).$$

We below show that $\mathcal{D} : 2^{S^c} \to \mathbb{R}$ is non-negative, monotone, and submodular, thus implying that $\boldsymbol{\mathcal{P}2}$ becomes a monotone submodular maximization problem. While it is straightforward to see that $\mathcal{D}$ is non-negative, it is non-trivial to show its monotonicity and submodularity.

Fix $W \subseteq S^c$. For some $u \in S^c \setminus W$, let $W^+ \triangleq W \cup \{u\}$. We define the marginal gain by

$$\Delta(W, u) \triangleq \mathcal{D}(W^+) - \mathcal{D}(W) = \mathcal{V}(W) - \mathcal{V}(W^+), \quad (12)$$

where the equality follows from (11). To show the monotonicity and submodularity of $\mathcal{D}$, we need to demonstrate that the marginal gain $\Delta(W, u)$ is non-negative and diminishes with increasing size of W. The technical challenge, however, is how to evaluate the quantities $\mathcal{V}(W)$ and $\mathcal{V}(W^+)$, which are *not directly comparable*, since they are defined on different graphs $G_W = (N, E_W)$ and $G_{W^+} = (N, E_{W^+})$, respectively. To overcome this challenge, we carefully define their corresponding quantities on the baseline graph G_S so that they can

Algorithm 1: Greedy Algorithm

```
Input  : $G_S = (N, E_S), S, b$
Output : $W$
1  $W \leftarrow \emptyset, G_W \leftarrow G_S$
2  while $|W| \leq b$ do
3      Compute $\mathcal{V}(W)$
4      for $u \in S^c \setminus W$ do
5          $W_+ \leftarrow W \cup \{u\}$
6          $E_{W^+} \leftarrow E_W \setminus \{(u, v) \in E_W \mid v \in S^c\}$
7          $G_{W^+} \leftarrow (N, E_{W^+})$
8          Compute $\mathcal{V}(W^+)$
9          $\Delta(W, u) = \mathcal{V}(W) - \mathcal{V}(W^+)$
10     $u^* \leftarrow \arg\max_{u \in S^c \setminus W} \Delta(W, u)$
11     $W \leftarrow W \cup \{u^*\}$
12     $E_W \leftarrow E_W \setminus \{(u^*, v) \in E_W \mid v \in S^c\}$
13     $G_W \leftarrow (N, E_W)$
```

be compared on the same ground, as detailed in the proof of the following.

Theorem 2. *$\mathcal{D}$ is a non-negative monotone submodular function.*

Proof. See our technical report [28]. □

Thanks to this nice property of $\mathcal{D}$, we are able to build a greedy algorithm for obtaining a solution W to $\mathcal{P}2$, as depicted in Algorithm 1. For any given budget b on the size of traps W, it is essentially finding an element u^* that maximizes the marginal gain $\Delta(W, u)$ every iteration until the size of the resulting set W becomes b. Therefore, this algorithm naturally achieves the following performance guarantee.

Corollary 1. *Algorithm 1 achieves a $(1-1/e)$-approximation to the optimal solution of $\mathcal{P}2$.*

Proof. For any given non-negative monotone submodular function f, which is $\mathcal{D}$ in our case, let W be a set of size b obtained in a greedy fashion, by choosing an element that provides the largest marginal gain in the value of f each time. Let $W^\star$ be a set maximizing the value of f over all b-element sets. It is well-known that $f(W) \geq (1 - 1/e) f(W^\star)$ [19, 32]. Thus, the result follows. □

4.2 Monte Carlo Estimation

While our greedy algorithm in Algorithm 1 can effectively solve $\mathcal{P}2$, there is still a computational issue with the marginal gain $\Delta(W, u)$, or $\mathcal{V}(W)$ for each W. Specifically, as can be seen from (9), it remains questionable how to efficiently compute $\sum_{t<\infty} q_t h_t(s, i; G_W)$. To address this issue, we below propose an MC estimator of this quantity based on R *i.i.d.* realizations (or sample paths) of a simple random walk of fixed length l on G_W. We also establish an (ϵ, δ)-approximation of this estimator, which implies, for any small $\epsilon, \delta > 0$, how many realizations R are necessary with a choice of l so that the approximation error can be bounded by ϵ with probability at least $1 - \delta$.

Fix $W \subset S^c$. For notational simplicity, we define

$$c(s, i) \triangleq \sum_{t<\infty} q_t h_t(s, i), \tag{13}$$

where we here drop G_W for $h_t(s, i; G_W)$. We also define by

$$c_l(s, i) \triangleq \sum_{t=1}^{l} q_t h_t(s, i), \tag{14}$$

the l-truncated version of $c(s, i)$, where $h_0(s, i) = 0$. Let $X^{(r)}(t)$, $r = 1, 2, \ldots, R$, be the position at time t of the r-th random-walk realization on G_W, assuming that $X^{(r)}(0) = s$ for each r. We then define, for all r and $t > 0$,

$$f_i\big(X^{(r)}(0), X^{(r)}(1), \ldots, X^{(r)}(t)\big) \triangleq \mathbb{1}\{X^{(r)}(t) = i, X^{(r)}(t') \neq i \text{ for all } t' < t\},$$

i.e., an indicator function of the event that the walk visits i at time t for the first time, with

$$\mathbb{E}\left[f_i\big(X^{(r)}(0), X^{(r)}(1), \ldots, X^{(r)}(t)\big)\right] = h_t(s, i),$$

which is the probability that the walk visits i at time t for the first time, i.e., the first hitting time to i is t. We first construct an MC estimator of $h_t(s, i)$, for all $t > 0, s \in S, i \in S^c$, as

$$\hat{h}_{R,t}(s, i) \triangleq \frac{1}{R} \sum_{r=1}^{R} f_i\big(X^{(r)}(0), X^{(r)}(1), \ldots, X^{(r)}(t)\big).$$

That is, this estimator is, in essence, the sample mean of f_i over R *i.i.d.* realizations of t-length simple random walk. By the strong law of large numbers, we have

$$\hat{h}_{R,t}(s, i) \xrightarrow{\text{a.s.}} h_t(s, i), \text{ as } R \to \infty. \tag{15}$$

We then construct an MC estimator of $c_l(s, i)$ in (14), for all $l > 0, s \in S, i \in S^c$, as

$$\hat{c}_{R,l}(s, i) \triangleq \sum_{t=1}^{l} q_t \hat{h}_{R,t}(s, i).$$

In other words, this estimator $\hat{c}_{R,l}(s, i)$ is a weighted sum of l estimators $\hat{h}_{R,t}(s, i)$, $t = 1, 2, \ldots, l$, with coefficients q_t, which are built upon R *i.i.d.* realizations of simple random walk of length l. By (15) and the linearity of almost sure convergence, we have

$$\hat{c}_{R,l}(s, i) \xrightarrow{\text{a.s.}} c_l(s, i), \text{ as } R \to \infty.$$

While $\hat{c}_{R,l}(s, i)$ is an asymptotically consistent estimator of $c_l(s, i)$ in (14) for each l, we below demonstrate that this estimator, with a proper choice of l, can also be used to approximate $c(s, i)$ in (13). To proceed, we need the following.

Theorem 3 (Hoeffding's inequality [17]). *Let $Y_1, \ldots, Y_n$ be i.i.d. random variables such that $\mathbb{E}[Y_i] = \mu$ and $a \leq Y_i \leq b$. Then, for any $\epsilon > 0$,*

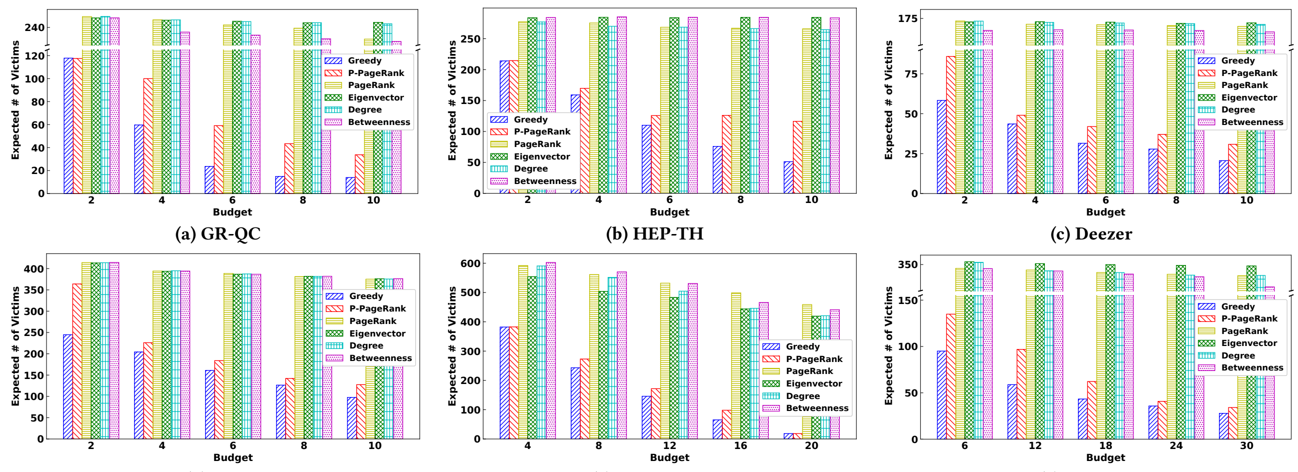
$$\mathbb{P}\left\{\left|\frac{1}{n}\sum_{i=1}^{n} Y_i - \mu\right| > \epsilon\right\} \leq 2e^{-2n\epsilon^2/(b-a)^2}.$$

We below show that $\hat{c}_{R,l}(s, i)$ achieves an (ϵ, δ)-approximation to $c(s, i)$ in (13), when l and R are properly chosen.

Theorem 4. *For any $\epsilon > 0$ and $\delta \in (0, 1)$, let l be chosen such that $\sum_{t=l+1}^{\infty} q_t \leq \frac{\epsilon}{2}$. If $R \geq \frac{2\mathbb{E}[L]^2}{\epsilon^2} \log(2l/\delta)$, then we have, for $s \in S, i \in S^c$,*

$$\mathbb{P}\left\{\left|\hat{c}_{R,l}(s, i) - c(s, i)\right| > \epsilon\right\} \leq \delta.$$

Proof. See our technical report [28]. □



(a) GR-QC (b) HEP-TH (c) Deezer (d) Digg (e) EmailEnron (f) Facebook

Figure 1: (Case 1) The expected number of victims with varying budget b.

Theorem 4 suggests a systematic way of choosing l and R to achieve provable guarantee on the accuracy of the estimator $\hat{c}_{R,l}(s,i)$ to approximate the quantity $c(s,i)$ for Algorithm 1, i.e., the approximation error is bounded by ϵ with probability at least $1-\delta$. In what follows, we numerically demonstrate that Algorithm 1 is effective and robust, even with possibly noisy estimates of $c(s,i)$ by $\hat{c}_{R,l}(s,i)$ for small l and R.

5 SIMULATION RESULTS

In this section, we present simulation results on various real-world graphs to demonstrate the efficacy of our greedy algorithm in Algorithm 1. We consider six real-world network datasets from the repositories such as SNAP [26] and KONECT [23], which allow us to test a wide range of network topologies with different formations of the adjacency matrix $\mathbf{A}$. In other words, we can evaluate the impact of diverse network topologies on the performance of our greedy algorithm. For simulations, we preprocess each graph to remove self-loops. We use the largest connected component of each graph to ensure graph connectivity. Their basic statistics are summarized in Table 1. For Digg, its undirected version is used.

Table 1: Graph statistics

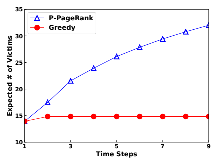
	# Nodes	# Edges	Avg. Degree	Diameter
GR-QC	4158	13422	6.456	17
HEP-TH	8638	24806	5.743	17
Facebook	22470	170823	15.205	15
Deezer	28281	92752	6.559	21
Digg	29652	84781	5.718	12
EmailEnron	33696	180811	10.732	11

Simulation setup. We consider the random length L_k of each crawler to be independently drawn from a common geometric distribution with parameter α, i.e., $\mathbb{P}\{L=t\}=(1-\alpha)^t\alpha$, $t=0,1,\ldots$, with $\mathbb{E}[L]=(1-\alpha)/\alpha$. We use $\alpha=0.1$, unless otherwise specified. We consider the following test cases for generating crawlers in each simulation. For both cases, each crawler independently starts from a node s chosen uniformly at random from a given source set S.

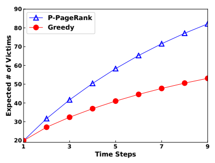
- *Case 1*: $K=100$ crawlers are launched *simultaneously* at the beginning of each simulation, which lasts until the lifetimes of all the crawlers are elapsed.
- *Case 2*: Crawlers are generated *at different times* according to a Poisson process with rate $1/\mathbb{E}[L]$ for 10^5 time slots in each simulation. To be precise, the inter-arrival time between two consecutive crawlers is exponentially distributed with the mean equal to $\mathbb{E}[L]$. Since time is divided into discrete slots, each inter-arrival time is rounded up to the nearest integer value that determines which time slot a next crawler is generated. In other words, a new crawler is launched roughly every $\mathbb{E}[L]$ time slots, which is the average lifetime of each crawler. Each simulation runs for a simulation time of 10^5 slots.

We also consider two different cases of choosing the source set S for a given size $|S|$. One is a *'cluster'* case, where all nodes in S are connected and form a cluster, and the cluster is chosen randomly from the graph, as will be explained below in detail. Note that this does not necessarily mean that S forms a clique, i.e., nodes in S may not be direct/one-hop neighbors of each other, but they are still connected via intermediate nodes. The other is a *'scatter'* case, where nodes in S are independently chosen uniformly at random from the node set N and thus they are likely disconnected and dispersed over the graph. Every data point reported here is obtained by taking the average over 10^4 independent simulations.

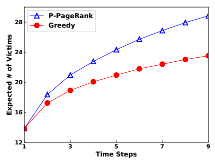
For a given source set S and a given budget b on the size of W, to evaluate the performance of our greedy algorithm in Algorithm 1, we consider the standard centrality measures such as degree centrality, betweenness centrality, eigenvector centrality, and PageRank [1, 33], as baseline algorithms. Since they measure the importance of each node in the graph, they can be efficient heuristics for $\boldsymbol{\mathcal{P}2}$ by choosing the top-b central nodes for the traps W. While they are agnostic to the source set S, we also consider Personalized PageRank [18] as a source-aware baseline algorithm. For Personalized PageRank, we set its damping factor (or teleportation probability) to be $1-\alpha$ or its default value of 0.85 in [18], and its preference vector to be $\mathbf{u}_S$, i.e., a uniform distribution over S. For the execution of Algorithm 1, we use the MC estimator $\hat{c}_{R,l}(s,i)$ with $l=80$ and $R=4\times10^4$ to compute $c(s,i)$ in (13) and eventually $\mathcal{V}(W)$ in (9) for each W. We also tested the case of $l=60$ and $R=2\times10^4$ for $\hat{c}_{R,l}(s,i)$, and observed almost the same results of our algorithm. We thus omit them for brevity.



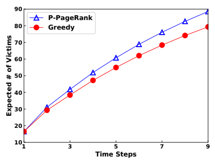
(a) GR-QC



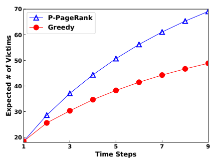
(b) HEP-TH



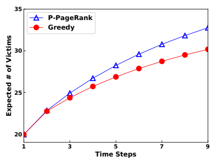
(c) Deezer



(d) Digg

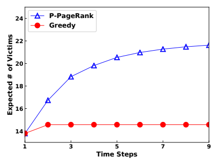


(e) EmailEnron

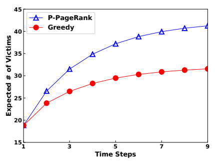


(f) Facebook

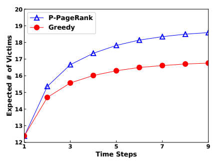
Figure 2: (Case 1) The expected number of victims over time.



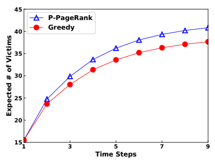
(a) GR-QC



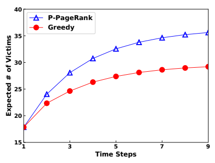
(b) HEP-TH



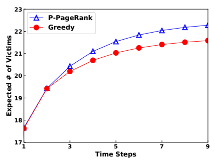
(c) Deezer



(d) Digg

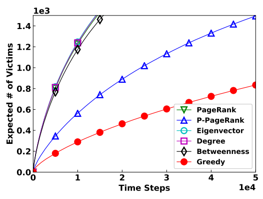


(e) EmailEnron

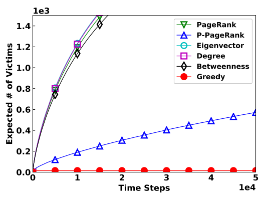


(f) Facebook

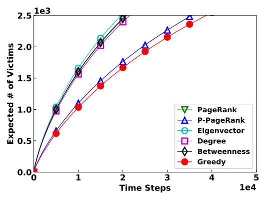
Figure 3: (Case 1) The expected number of victims over time, when $\alpha = 0.3$.



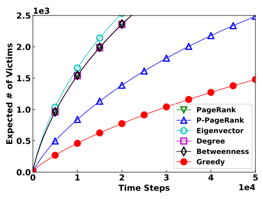
(a) GR-QC, $b = 4$



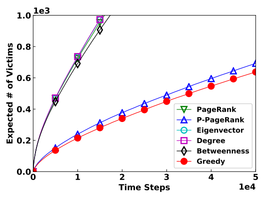
(b) GR-QC, $b = 8$



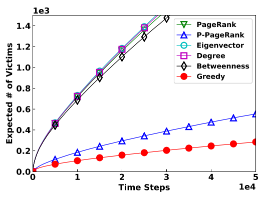
(c) HEP-TH, $b = 4$



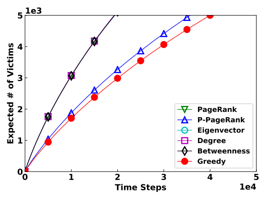
(d) HEP-TH, $b = 8$



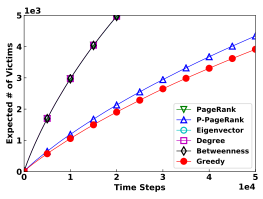
(e) Deezer, $b = 4$



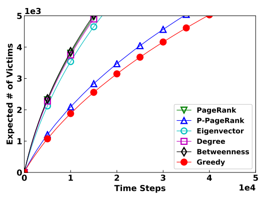
(f) Deezer, $b = 8$



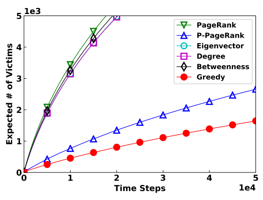
(g) Digg, $b = 4$



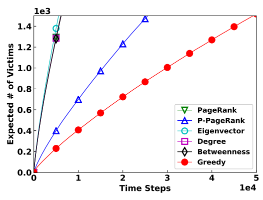
(h) Digg, $b = 8$



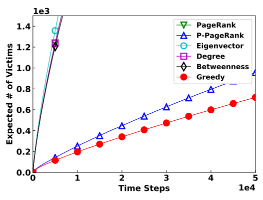
(i) EmailEnron, $b = 8$



(j) EmailEnron, $b = 16$



(k) Facebook, $b = 12$

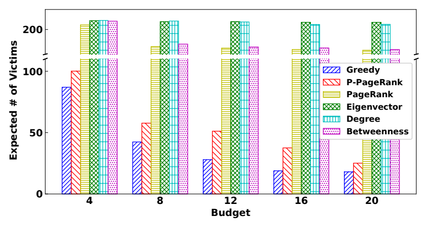


(l) Facebook, $b = 24$

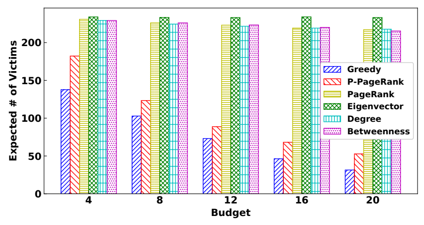
Figure 4: (Case 2) The expected number of cumulative victims over time.

Simulation results. We first present simulation results for Case 1 with the 'cluster' case, where the source-set/cluster size $|S|$ for each graph is set to be roughly the same as its average degree. Specifically, the cluster sizes are $|S| = 5$ for GR-QC, HEP-TH, Deezer and Digg, $|S| = 10$ for EmailEnron, and $|S| = 15$ for Facebook, respectively. For each graph, the source set S is selected by randomly choosing a node among the ones having more (one-hop) neighbors than $|S|$ and then including its $|S|$–1 neighbors. We here report the results on the performance of Personalized PageRank with its damping factor of 0.9. We also observed the same results when the damping factor is set to be the default value of 0.85, and omit them for brevity.

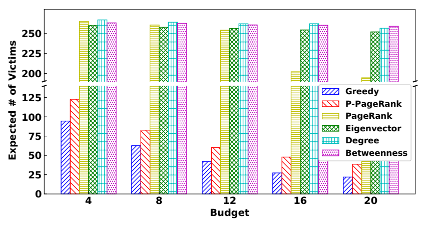
In Figure 1, we report the expected number of victims by K crawlers (measured at the end of simulation), where the victims are counted only once, when we change the budget b for the size of W up to twice the source-set size $|S|$. Our greedy algorithm in Algorithm 1 outperforms (no worse than) other baseline algorithms for all test cases and all graphs. We also observe that the performance improvement by Algorithm 1 compared to the baseline algorithms generally becomes more drastic as we increase the budget b. Another noticeable observation is that the performance of source-agnostic centrality measures is not effective as much as the source-aware algorithms, i.e., Personalized PageRank and Algorithm 1, and remains ineffective even with increasing the budget b. While our greedy algorithm generally performs better than Personalized PageRank, there are few cases where they exhibit very similar performance. The cases are when the budget is too small, i.e., $b = 2$ for GR-QC and HEP-TH, and $b = 4$ for EmailEnron, in which case their resulting sets W happen to be the same, and when the budget is too large, i.e., $b = 20$ for EmailEnron, in which case their resulting sets W happen to be overlapped with common



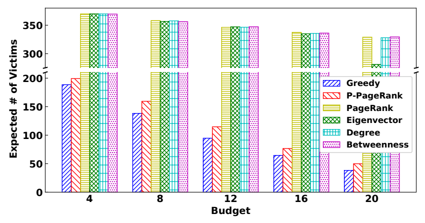
(a) GR-QC



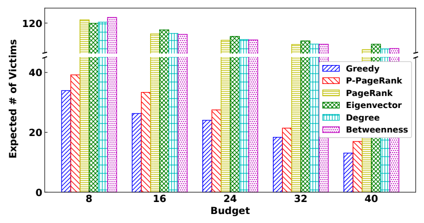
(b) HEP-TH



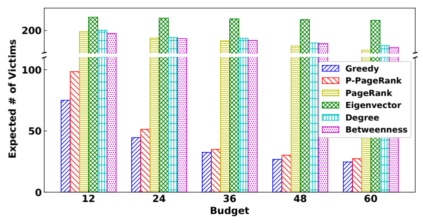
(c) Deezer



(d) Digg

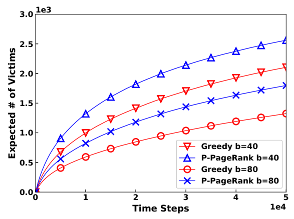


(e) EmailEnron

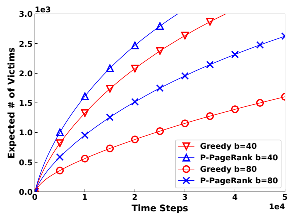


(f) Facebook

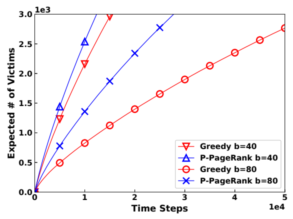
Figure 5: (Case 1) The expected number of victims with varying budget b, when the cluster size $|S|$ increases by a factor of two.



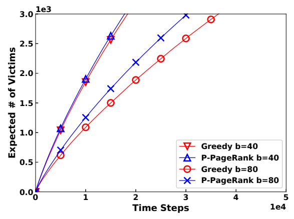
(a) GR-QC, $|S| = n \cdot 1\%$



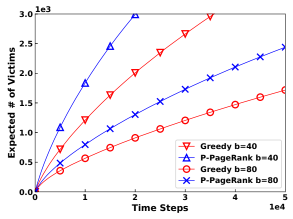
(b) HEP-TH, $|S| = n \cdot 0.5\%$



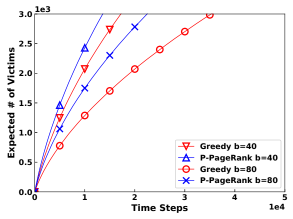
(c) Deezer, $|S| = n \cdot 0.1\%$



(d) Digg, $|S| = n \cdot 0.1\%$



(e) EmailEnron, $|S| = n \cdot 0.1\%$



(f) Facebook, $|S| = n \cdot 0.1\%$

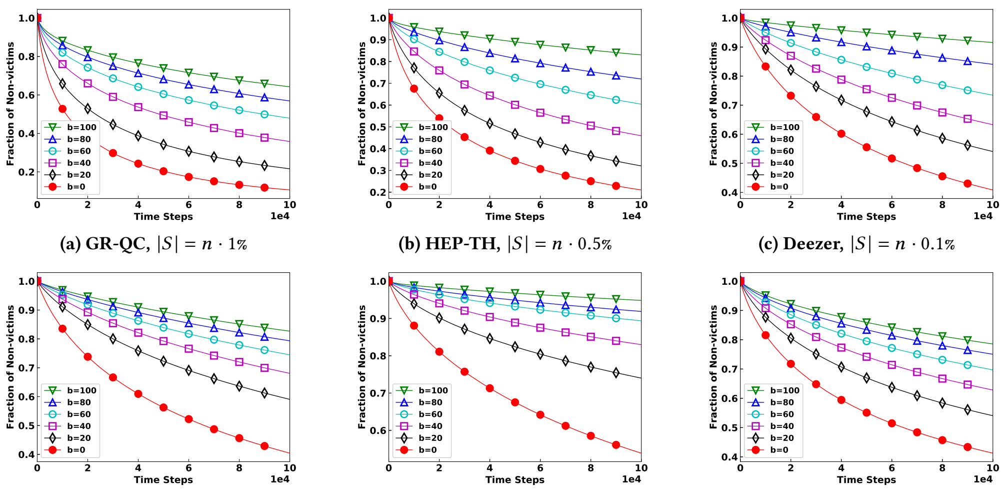
Figure 6: (Case 2) The expected number of cumulative victims over time, when the choice of S is the scatter case.

nodes/traps being the most effective ones in reducing the expected number of victims.

We further examine how the expected number of unique victims changes *over time*. Figure 2 shows the time-varying behavior when the budget b is the one with the fourth instance of each sub-figure in Figure 1, i.e., $b=8$ for GR-QC, HEP-TH, Deezer and Digg, $b=16$ for EmailEnron, and $b=24$ for Facebook, respectively. We here only report the results of our greedy algorithm and Personalized PageRank for brevity, since they perform much better than the other baseline algorithms. The results show that the superiority of our greedy algorithm over Personalized PageRank remains *intact* over time. In addition, we evaluate the impact of having different statistical properties of the random length L of each crawler, especially when the value of the parameter α of the geometric distribution changes. We consider $\alpha=0.3$. Note that the larger value of α makes the length of each crawler more likely to be shorter with the smaller expected length. As shown in Figure 3, which is obtained under the same setting as the one for Figure 2 but when $\alpha=0.3$, the expected number of victims becomes smaller as compared to that with $\alpha=0.1$, for each algorithm with the same budget b. While we again only report the results of our greedy algorithm and Personalized PageRank in Figure 3, we observe that our greedy algorithm is superior to other baseline algorithms, including Personalized PageRank.

We next present simulation results for Case 2 with the cluster source in Figure 4, where the cluster sizes remain the same as Case 1 for Figure 1, along with two cases of budget b, which correspond to the second and fourth instances of Figure 1, respectively. We report the expected number of *cumulative* (unique) victims over time, since crawlers are *continuously generated* for the entire time of each simulation. As shown in Figure 4, we first observe that there is no significant difference among the source-agnostic centrality measures, which are degree centrality, betweenness centrality, eigenvector centrality and PageRank, and the impact of increasing the budget b on their performance is still not much noticeable. We also observe that our greedy algorithm remains the best.

Figure 5 shows the simulation results of Case 1 with the cluster source when the cluster-source size $|S|$ now increases by a factor of two compared to the one for Figure 1. Specifically, the cluster sizes are $|S| = 10$ for GR-QC, HEP-TH, Deezer and Digg, $|S| = 20$ for EmailEnron, and $|S| = 30$ for Facebook, respectively. For each



(a) GR-QC, $|S| = n \cdot 1\%$ (b) HEP-TH, $|S| = n \cdot 0.5\%$ (c) Deezer, $|S| = n \cdot 0.1\%$

(d) Digg, $|S| = n \cdot 0.1\%$ (e) EmailEnron, $|S| = n \cdot 0.1\%$ (f) Facebook, $|S| = n \cdot 0.1\%$

Figure 7: (Case 2) A fraction of non-victims over time for Algorithm 1, when the choice of S is the scatter case.

graph, the set S is selected by randomly choosing a node among the nodes, each of whose number of one-hop and two-hop neighbors is greater than $|S|$, and then including $|S|-1$ of them (one-hop neighbors added first and then two-hop neighbors). The budget b also varies up to twice the size $|S|$. The results again manifest the superiority of our greedy algorithm over other baseline algorithms, including Personalized PageRank, while exhibiting similar trends as in Figure 1.

We finally consider Case 2 with the 'scatter' source, where the nodes in S are randomly chosen, and the source-set sizes are chosen to be roughly the same for all graphs. We only report the results of our greedy algorithm and Personalized PageRank in Figure 6, since they significantly outperform the other algorithms. Here we consider $b = 40$ and $b = 80$, which are about the same as the source-set size $|S|$ and twice as big as $|S|$, respectively. The results again confirm the superiority of our algorithm. We also evaluate the performance of our greedy algorithm in Algorithm 1 while changing the budget b. Figure 7 depicts how a fraction of non-victims over the nodes outside S changes over time. We observe that the amount of benefit with higher budget b depends on the underlying graph, and it decreases for certain graphs such as Digg, EmailEnron, and Facebook.

6 CONCLUSION

We have studied the problem of finding where to place a set of traps on a general social graph to catch adversarial crawlers with malicious intents, to minimize the expected number of their victims. We were able to show that the problem is essentially a monotone submodular maximization problem and thus developed a $(1-1/e)$-approximation greedy algorithm. Furthermore, we developed an MC estimator for the key quantity $\mathcal{V}(W)$ of the greedy algorithm to make it more computationally efficient and scalable for large graphs. Extensive simulation results on various real-world graphs have demonstrated the versatile properties of our greedy algorithm, including its superior performance over other baseline algorithms based on centrality measures.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Grant Nos. IIS-1908375 and IIS-1910749. C. Lee is the corresponding author.

A PROOF OF THEOREM 1

Given an undirected, connected graph $G(N', E')$ and an integer B, which form an instance of the VERTEX COVER decision problem, the problem asks if there exists a subset $W \subseteq N'$ of size at most B such that, for every $(i, j) \in E'$, $i \in W$ or $j \in W$. Equivalently, it asks if there is a vertex cover W of size exactly B.[§] Let $|N'| = n'$. In addition, an instance of the decision problem of $\boldsymbol{\mathcal{P}1}$ consists of a graph $G_S = G(N, E_S)$ with a source set $S \subset N$, an integer b, and a real number η. Given an instance, our decision problem asks if there exists a subset $W \subseteq S^c = N \setminus S$ of size b such that $\mathcal{V}(W) \geq \eta$.

Without loss of generality, we assume that $K = 1$. For any given instance of the VERTEX COVER decision problem with $G(N', E')$ and B, with any arbitrary choice of S, we can construct an instance of our decision problem by setting $N := S \cap N'$ and $E_S := E' \cup \{(i, j) | \, i \in S, j \in N'\}$, which is a set of edges that include E' and an outgoing edge from each node in S to every node in N', forming a graph $G_s = G(N, E_S)$, along with $b := B$ and $\eta := q_1 + q_2 \left(1 - \frac{b}{n'}\right)$. We below show that W is a solution of the VERTEX COVER decision problem if and only if it is a solution to our decision problem with $\mathcal{V}(W) \geq q_1 + q_2 \left(1 - \frac{b}{n'}\right)$, where $S^c = N'$ with $|S^c| = n'$.

On one hand, assume that W is a vertex cover. Observe that starting from $s \in S$, any random-walk crawler of length $l \geq 1$ moves to either $i \in W$ or $i \in S^c \setminus W$ in the first step. For the former, it is simply trapped in W in the first step. For the latter, it is always caught by W in the second step, since all neighbors of $i \in S^c \setminus W$ belong to the vertex cover W. Thus, from (9), we have

$$\mathcal{V}(W) = q_1 \sum_{i \in S^c} h_1(\mathbf{u}_S, i; G_W) + q_2 \sum_{i \in S^c} h_2(\mathbf{u}_S, i; G_W)$$

[§]If there is a vertex cover of size less than or equal to B, then there is also one of size exactly B.

$$= q_1 + q_2 \left(1 - \frac{b}{n'}\right). \quad (16)$$

The first term in the RHS of (16) follows from that

$$\sum_{i \in S^c} h_1(\mathbf{u}_S, i; G_W) = \frac{1}{|S|} \sum_{s \in S} \sum_{i \in S^c} h_1(s, i; G_W)$$
$$= \frac{1}{|S|} \sum_{s \in S} \sum_{i \in S^c} P_{si} = \frac{1}{|S|} \sum_{s \in S} \sum_{i \in S^c} \frac{1}{|S^c|} = 1,$$

from which each s has an outgoing edge to every $i \in S^c$. The second term in the RHS of (16) also follows from that

$$\sum_{i \in S^c} h_2(\mathbf{u}_S, i; G_W) = \frac{1}{|S|} \sum_{s \in S} \sum_{j \in W} h_2(s, i; G_W)$$
$$= \frac{1}{|S|} \sum_{s \in S} \sum_{j' \in S^c \setminus W} P_{sj'} \cdot 1 = 1 - \frac{b}{n'}.$$

On the other hand, suppose that W is not a vertex cover. There always exists a pair of two 'uncovered' nodes u and v, which are neighbors of each other and $u, v \notin W$. It implies that starting from $s \in S$, any crawler of length $l \geq 2$ can visit u and then v consecutively, or vice versa. The walk is then trapped in W in the next step or further explores new nodes before it is trapped later, if $l \geq 3$, since the nodes in $S^c = N'$ are all reachable from each other. Therefore, we have

$$\mathcal{V}(W) \geq \sum_{t=1}^{3} q_t \sum_{i \in S^c} h_t(\mathbf{u}_S, i; G_W) > q_1 + q_2 \left(1 - \frac{b}{n'}\right),$$

which is from (16).

REFERENCES

[1] A.-L. Barabási. 2016. *Network Science.* Cambridge University Press.

[2] J. Bonneau, J. Anderson, and G. Danezis. 2009. Prying data out of a social network. In *Proceedings of IEEE/ACM ASONAM.* 249–254.

[3] P. Brémaud. 1999. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues.* Springer-Verlag.

[4] B. Carminati, E. Ferrari, and A. Perego. 2009. Enforcing Access Control in Web-Based Social Networks. *ACM Trans. Inf. Syst. Secur.* 13, 1, Article 6 (Nov. 2009).

[5] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. 2008. Epidemic Thresholds in Real Networks. *ACM Trans. Inf. Syst. Secur.* 10, 4, Article 1 (2008).

[6] H. Chan, L. Akoglu, and H. Tong. 2014. Make It or Break It: Manipulating Robustness in Large Networks. In *Proceedings of SIAM International Conference on Data Mining.* 325–333.

[7] C. Chen, H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. 2016. Eigen-Optimization on Large Graphs by Edge Manipulation. *ACM Trans. Knowl. Discov. Data* 10, 4, Article 49 (June 2016).

[8] C. Chen, H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. 2016. Node Immunization on Large Graphs: Theory and Algorithms. *IEEE Trans. Knowl. Data Eng.* 28, 1 (Jan. 2016), 113–126.

[9] X. Chen, Y. Li, P. Wang, and J. C. S. Lui. 2016. A general framework for estimating graphlet statistics via random walk. *Proceedings of the VLDB Endowment* 10, 3 (Nov. 2016), 253–264.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. 2009. *Introduction to Algorithms.* MIT press.

[11] G. Danezis and B. Wittneben. 2006. The Economics of Mass Surveillance and the Questionable Value of Anonymous Communications. In *Proceedings of the 5th Workshop on the Economics of Information Security.*

[12] A. Ganesh, L. Massoulié, and D. Towsley. 2005. The Effect of Network Topology on the Spread of Epidemics. In *Proceedings of IEEE INFOCOM.* 1455–1466.

[13] C. Gao, J. Liu, and N. Zhong. 2011. Network immunization and virus propagation in email networks: experimental evaluation and analysis. *Knowledge and Information Systems* 27, 2 (2011), 253–279.

[14] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. 2010. Walking in Facebook: A case study of unbiased sampling of OSNs. In *Proc. IEEE INFOCOM.* 1–9.

[15] G. Golnari, Y. Li, and Z.-L. Zhang. 2015. Pivotality of nodes in reachability problems using avoidance and transit hitting time metrics. In *Proceedings of International Conference on World Wide Web.* 1073–1078.

[16] X. He, G. Song, W. Chen, and Q. Jiang. 2012. Influence blocking maximization in social networks under the competitive linear threshold model. In *Proceedings of SIAM International Conference on Data Mining.* 463–474.

[17] W. Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *J. Amer. Statist. Assoc.* 58, 301 (1963), 13–30.

[18] G. Jeh and J. Widom. 2003. Scaling personalized web search. In *Proceedings of International Conference on World Wide Web.* 271–279.

[19] D. Kempe, J. Kleinberg, and É. Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of ACM SIGKDD.* 137–146.

[20] E. B. Khalil, B. Dilkina, and L. Song. 2014. Scalable Diffusion-Aware Optimization of Network Topology. In *Proceedings of ACM SIGKDD.* 1226–1235.

[21] M. Kimura, K. Saito, and H. Motoda. 2009. Blocking Links to Minimize Contamination Spread in a Social Network. *ACM Trans. Knowl. Discov. Data* 3, 2, Article 9 (April 2009).

[22] C. J. Kuhlman, G. Tuli, S. Swarup, M. V. Marathe, and S. S. Ravi. 2013. Blocking simple and complex contagion by edge removal. In *Proc. IEEE ICDM.* 399–408.

[23] J. Kunegis. 2013. KONECT – The Koblenz Network Collection. In *Proceedings of International Conference on World Wide Web.* 1343–1350.

[24] Kaspersky Lab. [n. d.]. What is a honeypot? https://www.kaspersky.com/resource-center/threats/what-is-a-honeypot.

[25] C.-H. Lee, X. Xu, and D. Y. Eun. 2012. Beyond random walk and Metropolis-Hastings samplers: Why you should not backtrack for unbiased graph sampling. In *Proceedings of ACM SIGMETRICS.* 319–330.

[26] J. Leskovec and A. Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[27] R. Li, J. X. Yu, X. Huang, and H. Cheng. 2014. Random-walk domination in large graphs. In *Proceedings of IEEE ICDE.* 736–747.

[28] S. Li, C.-H. Lee, and D. Y. Eun. 2020. *Trapping Malicious Crawlers in Social Networks.* Technical Report.

[29] J. Lindamood, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham. 2009. Inferring Private Information using Social Network Data. In *Proceedings of International Conference on World Wide Web.* 1145–1146.

[30] C. Mavroforakis, M. Mathioudakis, and A. Gionis. 2015. Absorbing random-walk centrality: Theory and algorithms. In *Proceedings of IEEE ICDM.* 901–906.

[31] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. 2007. Measurement and analysis of online social networks. In *Proceedings of ACM SIGCOMM Conference on Internet Measurement.* 29–42.

[32] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming* 14, 1 (1978), 265–294.

[33] M. E. J. Newman. 2010. *Networks: An Introduction.* Oxford University Press.

[34] M. E. J. Newman and M. Girvan. 2004. Finding and evaluating community structure in networks. *Phys. Rev. E* 69 (Feb 2004), 026113. Issue 2.

[35] H. T. Nguyen, A. Cano, T. Vu, and T. N. Dinh. 2020. Blocking Self-Avoiding Walks Stops Cyber-Epidemics: A Scalable GPU-Based Approach. *IEEE Trans. Knowl. Data Eng.* 32, 7 (2020), 1263–1275.

[36] J. R. Norris. 1997. *Markov Chains.* Cambridge University Press.

[37] C. Reuter and S. Scholl. 2014. Technical Limitations for Designing Applications for Social Media. In *Proceedings of Mensch & Computer 2014 – Workshopband.*

[38] S. Saha, A. Adiga, B. A. Prakash, and A. K. S. Vullikanti. 2015. Approximation Algorithms for Reducing the Spectral Radius to Control Epidemic Spread. In *Proceedings of SIAM International Conference on Data Mining.* 568–576.

[39] C. M. Schneider, T. Mihaljev, S. Havlin, and H. J. Herrmann. 2011. Suppressing epidemics with a limited amount of immunization units. *Phys. Rev. E* 84 (Dec 2011), 061911. Issue 6.

[40] Techopedia. [n. d.]. What is a Spider Trap? https://www.techopedia.com/definition/5197/spider-trap.

[41] P. Van Mieghem, D. Stevanović, F. Kuipers, C. Li, R. van de Bovenkamp, D. Liu, and H. Wang. 2011. Decreasing the spectral radius of a graph by link removals. *Phys. Rev. E* 84 (Jul 2011), 016101. Issue 1.

[42] N. Vesdapunt and H. Garcia-Molina. 2016. Updating an Existing Social Graph Snapshot via a Limited API. In *Proceedings of ACM CIKM.* 1693–1702.

[43] B. Viswanath et al. 2014. Towards detecting anomalous user behavior in online social networks. In *Proceedings of USENIX Security Symposium.* 223–238.

[44] P. Wang, J. Zhao, J. C. S. Lui, D. Towsley, and X. Guan. 2015. Unbiased Characterization of Node Pairs over Large Graphs. *ACM Trans. Knowl. Discov. Data* 9, 3, Article 22 (April 2015).

[45] S. Wasserman and K. Faust. 1994. *Social Network Analysis: Methods and Applications.* Cambridge University Press.

[46] C. Xiao, D. M. Freeman, and T. Hwa. 2015. Detecting clusters of fake accounts in online social networks. In *Proceedings of ACM AISec.* 91–101.

[47] W. Xu, X. Zhou, and L. Li. 2008. Inferring privacy information via social relations. In *Proceedings of IEEE ICDE Workshops.* 525–530.

[48] X. Xu, C.-H. Lee, and D. Y. Eun. 2017. Challenging the limits: Sampling online social networks with cost constraints. In *Proceedings of IEEE INFOCOM.* 1–9.

[49] E. Zhang, G. Wang, K. Gao, and G. Yu. 2015. Finding critical blocks of information diffusion in social networks. *World Wide Web* 18, 3 (2015), 731–747.

[50] Y. Zhu, Z. Hu, H. Wang, H. Hu, and G.-J. Ahn. 2010. A collaborative framework for privacy protection in online social networks. In *Proc. CollaborateCom.* 1–10.